# Объектіге-бағытталған бағдарламалау[[өңдеу](https://kk.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83&veaction=edit&section=0&summary=/*%20%D0%9A%D1%96%D1%80%D1%96%D1%81%D0%BF%D0%B5%20%D0%B1%D3%A9%D0%BB%D1%96%D0%BC%D1%96%D0%BD%20%D3%A9%D2%A3%D0%B4%D0%B5%D0%B4%D1%96%20*/%20) | [қайнарын өңдеу](https://kk.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83&action=edit&section=0&summary=/*%20%D0%9A%D1%96%D1%80%D1%96%D1%81%D0%BF%D0%B5%20%D0%B1%D3%A9%D0%BB%D1%96%D0%BC%D1%96%D0%BD%20%D3%A9%D2%A3%D0%B4%D0%B5%D0%B4%D1%96%20*/%20)]

Уикипедия — ашық энциклопедиясынан алынған мәлімет

[Навигацияға өту](https://kk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83#mw-head)[Іздеуге өту](https://kk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83#searchInput)

**Объектіге-бағытталған бағдарламалау** ([*орыс тілі*](https://kk.wikipedia.org/wiki/%D0%9E%D1%80%D1%8B%D1%81_%D1%82%D1%96%D0%BB%D1%96)*Объектно-ориентированное программирование;*[*ағыл.*](https://kk.wikipedia.org/wiki/%D0%90%D2%93%D1%8B%D0%BB%D1%88%D1%8B%D0%BD_%D1%82%D1%96%D0%BB%D1%96)*Object-orinted programming*) — объектілерді қолдану арқылы бағдарламалау парадигмасы, объектілік-бағдарланған архитектурасы пайдаланады. Мәліметтердің әрбір құрылымы тек бір программалық модульде тұруы тиіс. Бұл [мәліметтер](https://kk.wikipedia.org/wiki/%D0%9C%D3%99%D0%BB%D1%96%D0%BC%D0%B5%D1%82%D1%82%D0%B5%D1%80) құрылымындағы өзгерістердің және мәліметтермен орындалатын әрекеттердің оңашалануына кепілдік береді. Мәліметтерге кез келген қатынас құру тек модульді пайдалану арқылы болады. Объектілік-бағдарланған программалау негізгі 3 қасиетімен: *инкапсуляцияларымен*, *мұрагерлігімен* және *полиморфизмімен* сипатталады. Объект ұғымы дәстүрлі [Паскаль](https://kk.wikipedia.org/wiki/%D0%9F%D0%B0%D1%81%D0%BA%D0%B0%D0%BB%D1%8C), [Си процедуралық-бейімделген программалау](https://kk.wikipedia.org/w/index.php?title=%D0%A1%D0%B8_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%B0%D0%BB%D1%8B%D2%9B-%D0%B1%D0%B5%D0%B9%D1%96%D0%BC%D0%B4%D0%B5%D0%BB%D0%B3%D0%B5%D0%BD_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D0%BB%D0%B0%D1%83&action=edit&redlink=1) тілдеріне де енгізіледі.[[1]](https://kk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83#cite_note-1)



## Мазмұны

* [1Кіріспе](https://kk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83#%D0%9A%D1%96%D1%80%D1%96%D1%81%D0%BF%D0%B5)
  + [1.1Объектілі-бағытталған бағдарламалаудың негізгі принциптері:](https://kk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83#%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%BB%D1%96-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83%D0%B4%D1%8B%D2%A3_%D0%BD%D0%B5%D0%B3%D1%96%D0%B7%D0%B3)
* [2Тағы қараңыз](https://kk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83#%D0%A2%D0%B0%D2%93%D1%8B_%D2%9B%D0%B0%D1%80%D0%B0%D2%A3%D1%8B%D0%B7)
* [3Дереккөздер](https://kk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83#%D0%94%D0%B5%D1%80%D0%B5%D0%BA%D0%BA%D3%A9%D0%B7%D0%B4%D0%B5%D1%80)

## Кіріспе[[өңдеу](https://kk.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83&veaction=edit&section=1) | [қайнарын өңдеу](https://kk.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83&action=edit&section=1)]

*Объектіге-бағытталған бағдарламалау* – объектілердің көмегімен бағдарламаларды жобалау, жазу. Объектіге-бағытталған бағдарламалаудың негізі болып объект саналады. Объект деп – мәліметтер және олармен жұмыс істеу әдістерінің жиынтығы аталады. Объектіге-бағытталған бағдарлама деп – объектілер және олардың өзара әрекеттесу тәсілдерінің жиынтығы аталады.

##### Объектілі-бағытталған бағдарламалаудың негізгі принциптері:**[**[**өңдеу**](https://kk.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83&veaction=edit&section=2)**|**[**қайнарын өңдеу**](https://kk.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83&action=edit&section=2)**]**

1. Полиморфизм;
2. Инкапсуляция;
3. Мұрагерлік.

**Полиморфизм** - бір иерархиядағы объектілер үшін бір атаумен берілген әдістерді қолдану мүмкіндігі.  
**Инкапсуляция** - бір объектілік типте мәліметтерді біріктіру.  
**Мұрагерлік** - бір класстың мінездемелерін басқа класс арқылы қолдану мүмкінідігі.

## Тағы қараңыз[[өңдеу](https://kk.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83&veaction=edit&section=3) | [қайнарын өңдеу](https://kk.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83&action=edit&section=3)]

* [Объектіге-бағытталған бағдарламалау тілі](https://kk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83_%D1%82%D1%96%D0%BB%D1%96)
* [Java](https://kk.wikipedia.org/wiki/Java)
* [C Sharp](https://kk.wikipedia.org/wiki/C_Sharp)

## Дереккөздер[[өңдеу](https://kk.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83&veaction=edit&section=4) | [қайнарын өңдеу](https://kk.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83&action=edit&section=4)]

1. [↑](https://kk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83#cite_ref-1) Қазақ тілі терминдерінің салалық ғылыми түсіндірме сөздігі: Информатика және компьютерлік техника / Жалпы редакциясын басқарған – түсіндірме сөздіктер топтамасын шығару жөніндегі ғылыми-баспа бағдарламасының ғылыми жетекшісі, педагогика ғылымдарының докторы, профессор, Қазақстан Республикасы Мемлекеттік сыйлығының лауреаты А. Қ. Құсайынов. – Алматы: «Мектеп» баспасы» ЖАҚ, 2002 жыл. – 456 бет. [ISBN 5-7667-8284-5](https://kk.wikipedia.org/wiki/%D0%90%D1%80%D0%BD%D0%B0%D0%B9%D1%8B:%D0%9A%D1%96%D1%82%D0%B0%D0%BF_%D2%9B%D0%B0%D0%B9%D0%BD%D0%B0%D1%80%D0%BB%D0%B0%D1%80%D1%8B/5766782845)

# Объектіге-бағытталған бағдарламалау тілі[[өңдеу](https://kk.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83_%D1%82%D1%96%D0%BB%D1%96&veaction=edit&section=0&summary=/*%20%D0%9A%D1%96%D1%80%D1%96%D1%81%D0%BF%D0%B5%20%D0%B1%D3%A9%D0%BB%D1%96%D0%BC%D1%96%D0%BD%20%D3%A9%D2%A3%D0%B4%D0%B5%D0%B4%D1%96%20*/%20) | [қайнарын өңдеу](https://kk.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83_%D1%82%D1%96%D0%BB%D1%96&action=edit&section=0&summary=/*%20%D0%9A%D1%96%D1%80%D1%96%D1%81%D0%BF%D0%B5%20%D0%B1%D3%A9%D0%BB%D1%96%D0%BC%D1%96%D0%BD%20%D3%A9%D2%A3%D0%B4%D0%B5%D0%B4%D1%96%20*/%20)]

Уикипедия — ашық энциклопедиясынан алынған мәлімет

[Навигацияға өту](https://kk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83_%D1%82%D1%96%D0%BB%D1%96#mw-head)[Іздеуге өту](https://kk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83_%D1%82%D1%96%D0%BB%D1%96#searchInput)

**Объектіге-бағытталған бағдарламалау тілі** (ОБ-тіл) — [объектіге бағытталған бағдарламалау](https://kk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%BB%D1%96-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83) принциптеріне негізделген тіл.

Объектіге бағытталған бағдарламалаудың концепциясының негізінде объект түсінігі жатыр. Объект ол өрістер (мәліметтер) мен әдістерді (объект орындайтын қимылдар) біріктіретін субстанция болып табылады.

Мысалға, адам объектісі аты, туған жері өрістерін және жеу, ұйықтау, жүгіру әдістерін иемденеді. Соған сәйкес бағдарламада біз осындай операторларды қолдана аламыз: *Адам.Аты:="Ерлан";*және*Адам.Жеу(Тамақ);*



## Мазмұны

* [1Ерекшеліктері](https://kk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83_%D1%82%D1%96%D0%BB%D1%96#%D0%95%D1%80%D0%B5%D0%BA%D1%88%D0%B5%D0%BB%D1%96%D0%BA%D1%82%D0%B5%D1%80%D1%96)
* [2Бағдарламалау тілдерінің тізімі](https://kk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83_%D1%82%D1%96%D0%BB%D1%96#%D0%91%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83_%D1%82%D1%96%D0%BB%D0%B4%D0%B5%D1%80%D1%96%D0%BD%D1%96%D2%A3_%D1%82%D1%96%D0%B7%D1%96%D0%BC%D1%96)
* [3Әдебиеттер тізімі](https://kk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83_%D1%82%D1%96%D0%BB%D1%96#%D3%98%D0%B4%D0%B5%D0%B1%D0%B8%D0%B5%D1%82%D1%82%D0%B5%D1%80_%D1%82%D1%96%D0%B7%D1%96%D0%BC%D1%96)
* [4Сілтемелер](https://kk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83_%D1%82%D1%96%D0%BB%D1%96#%D0%A1%D1%96%D0%BB%D1%82%D0%B5%D0%BC%D0%B5%D0%BB%D0%B5%D1%80)

## Ерекшеліктері[[өңдеу](https://kk.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83_%D1%82%D1%96%D0%BB%D1%96&veaction=edit&section=1) | [қайнарын өңдеу](https://kk.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83_%D1%82%D1%96%D0%BB%D1%96&action=edit&section=1)]

Қазіргі замаңғы объектіге бағытталған бағдарламалау тілдерінде мына әдістер болады:

* [*Мұрагерлеу*](https://kk.wikipedia.org/w/index.php?title=%D0%9C%D2%B1%D1%80%D0%B0%D0%B3%D0%B5%D1%80%D0%BB%D0%B5%D1%83_(%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83)&action=edit&redlink=1). Қосымша элементтерді ([әдістерді](https://kk.wikipedia.org/wiki/%D3%98%D0%B4%D1%96%D1%81)) қосу арқылы жаңа объектілер классын жасау. Қазіргі таңда ОБ тілдері [көптік мұрагерлеуді](https://kk.wikipedia.org/w/index.php?title=%D0%9A%D3%A9%D0%BF%D1%82%D1%96%D0%BA_%D0%BC%D2%B1%D1%80%D0%B0%D0%B3%D0%B5%D1%80%D0%BB%D0%B5%D1%83&action=edit&redlink=1) қолдануға рұқсат береді, яғни бір класста бірнеше класстың мүмкіндіктерін біріктіреді.
* [*Инкапсуляция*](https://kk.wikipedia.org/w/index.php?title=%D0%98%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_(%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83)&action=edit&redlink=1). Бағдарламаның бір бөлігін басқа бөліктерге әсер етпей өзгертуді қамтамасыз ететін реализацияларды құпиялайды. Ол өз кезегінде бағдарламаның модификациясын және қолдауын әжептәуір жеңілдетеді.
* [*Полиморфизм*](https://kk.wikipedia.org/w/index.php?title=%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_(%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83)&action=edit&redlink=1). Полиформизм кезінде әкелік класстың кейбір бөліктері (әдістері) белгілі бір балалық класстың спецификалық әрекетін орындайтын жаңа әдіспен ауыстырылады. Бұл жағдайда кластың интерфейсі өзгеріссіз қалады, бірақ бір атаулы әдістердің орындалуы мен параметрлері өзгешеленеді. Полиформизммен соңғы қосылулар тығыз байланыста болады.
* [*Типизация*](https://kk.wikipedia.org/w/index.php?title=%D0%A2%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F&action=edit&redlink=1). [Компиляция](https://kk.wikipedia.org/w/index.php?title=%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%86%D0%B8%D1%8F&action=edit&redlink=1) кезінде көптеген қателерді жоюға мүмкіндік береді. Операциялар белгілі бір типтегі [объектілермен](https://kk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82) ғана орындалады.

## Бағдарламалау тілдерінің тізімі[[өңдеу](https://kk.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83_%D1%82%D1%96%D0%BB%D1%96&veaction=edit&section=2) | [қайнарын өңдеу](https://kk.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83_%D1%82%D1%96%D0%BB%D1%96&action=edit&section=2)]

Объектіге бағытталған бағдарламалау тілдерінің толық емес тізімі:

|  |  |  |
| --- | --- | --- |
| * [C#](https://kk.wikipedia.org/wiki/C_Sharp) * [Java](https://kk.wikipedia.org/wiki/Java) * [Delphi](https://kk.wikipedia.org/wiki/Delphi_(%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83_%D1%82%D1%96%D0%BB%D1%96)) * [Eiffel](https://kk.wikipedia.org/w/index.php?title=Eiffel&action=edit&redlink=1) * [Simula](https://kk.wikipedia.org/w/index.php?title=Simula&action=edit&redlink=1) * [D](https://kk.wikipedia.org/w/index.php?title=D_(%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83_%D1%82%D1%96%D0%BB%D1%96)&action=edit&redlink=1) * [Io](https://kk.wikipedia.org/w/index.php?title=Io&action=edit&redlink=1) * [Objective-C](https://kk.wikipedia.org/wiki/Objective-C) | * [Object Pascal](https://kk.wikipedia.org/w/index.php?title=Object_Pascal&action=edit&redlink=1) * [VB.NET](https://kk.wikipedia.org/w/index.php?title=Visual_Basic_.NET&action=edit&redlink=1) * [Visual DataFlex](https://kk.wikipedia.org/w/index.php?title=Visual_DataFlex&action=edit&redlink=1) * [Perl](https://kk.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%BB) * [PowerBuilder](https://kk.wikipedia.org/w/index.php?title=PowerBuilder&action=edit&redlink=1) * [Python](https://kk.wikipedia.org/wiki/Python) * [Scala](https://kk.wikipedia.org/w/index.php?title=Scala_(%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83_%D1%82%D1%96%D0%BB%D1%96)&action=edit&redlink=1) * [ActionScript](https://kk.wikipedia.org/w/index.php?title=ActionScript&action=edit&redlink=1) (3.0) | * [JavaScript](https://kk.wikipedia.org/wiki/JavaScript) * [JScript .NET](https://kk.wikipedia.org/w/index.php?title=JScript_.NET&action=edit&redlink=1) * [Ruby](https://kk.wikipedia.org/w/index.php?title=Ruby&action=edit&redlink=1) * [Smalltalk](https://kk.wikipedia.org/w/index.php?title=Smalltalk&action=edit&redlink=1) * [Ada](https://kk.wikipedia.org/wiki/%D0%90%D0%B4%D0%B0) * [Xbase++](https://kk.wikipedia.org/w/index.php?title=Xbase%2B%2B&action=edit&redlink=1) * [X++](https://kk.wikipedia.org/w/index.php?title=X%2B%2B&action=edit&redlink=1) * [Vala](https://kk.wikipedia.org/wiki/%D0%92%D0%B0%D0%BB%D0%B0) * [PHP](https://kk.wikipedia.org/wiki/PHP) |

Ортақ мақсатты бұл бағдарламалау тілдерінен басқа, специализациясы төмен тілдер де бар.

## Әдебиеттер тізімі[[өңдеу](https://kk.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83_%D1%82%D1%96%D0%BB%D1%96&veaction=edit&section=3) | [қайнарын өңдеу](https://kk.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%96%D0%B3%D0%B5-%D0%B1%D0%B0%D2%93%D1%8B%D1%82%D1%82%D0%B0%D0%BB%D2%93%D0%B0%D0%BD_%D0%B1%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83_%D1%82%D1%96%D0%BB%D1%96&action=edit&section=3)]

Объектіге бағытталған әдістер.Принципі мен практикасы. Авторы - Иан Грэхем, 3-ші басылым, ISBN = 0-201-61913-X

# Объектно-ориентированное программирование

[[править](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&veaction=edit&section=0) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&section=0&summary=/*%20%D0%9F%D1%80%D0%B5%D0%B0%D0%BC%D0%B1%D1%83%D0%BB%D0%B0%20*/%20)]

Материал из Википедии — свободной энциклопедии

Текущая версия страницы пока [не проверялась](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F:%D0%9F%D1%80%D0%BE%D0%B2%D0%B5%D1%80%D0%BA%D0%B0_%D1%81%D1%82%D0%B0%D1%82%D0%B5%D0%B9/%D0%9F%D0%BE%D1%8F%D1%81%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B4%D0%BB%D1%8F_%D1%87%D0%B8%D1%82%D0%B0%D1%82%D0%B5%D0%BB%D0%B5%D0%B9" \o "Википедия:Проверка статей/Пояснение для читателей) опытными участниками и может значительно отличаться от [версии](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&stable=1), проверенной 15 марта 2022 года; проверки требуют [40 правок](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&oldid=120651247&diff=cur&diffonly=0).

[Перейти к навигации](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#mw-head)[Перейти к поиску](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#searchInput)

|  |
| --- |
|  |
|  |

**Объе́ктно-ориенти́рованное**[**программи́рование**](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) ([сокр.](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D0%B1%D1%80%D0%B5%D0%B2%D0%B8%D0%B0%D1%82%D1%83%D1%80%D0%B0) ООП) — [методология программирования](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), основанная на представлении программы в виде совокупности взаимодействующих [объектов](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), каждый из которых является экземпляром определённого [класса](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), а классы образуют [иерархию](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F) [наследования](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))[[1]](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#cite_note-_ce7b104dbaa5d933-1).

Идеологически, ООП — подход к программированию как к [моделированию](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) информационных объектов, решающий на новом уровне основную задачу [структурного программирования](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5): структурирование информации с точки зрения управляемости[[2]](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#cite_note-2), что существенно улучшает управляемость самим процессом моделирования, что, в свою очередь, особенно важно при реализации крупных проектов.

Управляемость для иерархических систем предполагает минимизацию избыточности данных (аналогичную [нормализации](https://ru.wikipedia.org/wiki/%D0%9D%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D0%B0)) и их целостность, поэтому созданное удобно управляемым — будет и удобно пониматься. Таким образом, через тактическую задачу управляемости решается стратегическая задача — транслировать понимание задачи программистом в наиболее удобную для дальнейшего использования форму.

Основные [принципы](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF) структурирования в случае ООП связаны с различными аспектами базового понимания предметной задачи, которое требуется для оптимального управления соответствующей моделью:

* [абстракция](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%86%D0%B8%D1%8F_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) для выделения в моделируемом предмете важного для решения конкретной задачи по предмету, в конечном счёте — контекстное понимание предмета, формализуемое в виде класса;
* [инкапсуляция](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) для быстрой и безопасной организации собственно иерархической управляемости: чтобы было достаточно простой команды «что делать», без одновременного уточнения как именно делать, так как это уже другой уровень управления;
* [наследование](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) для быстрой и безопасной организации родственных понятий: чтобы было достаточно на каждом иерархическом шаге учитывать только изменения, не дублируя всё остальное, учтённое на предыдущих шагах;
* [полиморфизм](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) для определения точки, в которой единое управление лучше распараллелить или наоборот — собрать воедино.

То есть фактически речь идёт о прогрессирующей организации информации согласно первичным семантическим критериям: «важное/неважное», «ключевое/подробности», «родительское/дочернее», «единое/множественное». Прогрессирование, в частности, на последнем этапе даёт возможность перехода на следующий уровень детализации, что замыкает общий процесс.

Обычный человеческий язык в целом отражает идеологию ООП, начиная с инкапсуляции представления о предмете в виде его имени и заканчивая полиморфизмом использования слова в переносном смысле, что в итоге развивает[[3]](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#cite_note-3) выражение представления через имя предмета до полноценного понятия-класса.



## Содержание

* [1Основные понятия](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%8B%D0%B5_%D0%BF%D0%BE%D0%BD%D1%8F%D1%82%D0%B8%D1%8F)
* [2Классификация подвидов ООП](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#%D0%9A%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F_%D0%BF%D0%BE%D0%B4%D0%B2%D0%B8%D0%B4%D0%BE%D0%B2_%D0%9E%D0%9E%D0%9F)
* [3История](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#%D0%98%D1%81%D1%82%D0%BE%D1%80%D0%B8%D1%8F)
* [4Определение ООП и его основные концепции](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#%D0%9E%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%9E%D0%9E%D0%9F_%D0%B8_%D0%B5%D0%B3%D0%BE_%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%8B%D0%B5_%D0%BA%D0%BE%D0%BD%D1%86%D0%B5%D0%BF%D1%86%D0%B8%D0%B8)
  + [4.1Сложности определения](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#%D0%A1%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D0%B8_%D0%BE%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F)
  + [4.2Концепции](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#%D0%9A%D0%BE%D0%BD%D1%86%D0%B5%D0%BF%D1%86%D0%B8%D0%B8)
* [5Особенности реализации](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#%D0%9E%D1%81%D0%BE%D0%B1%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D0%B8_%D1%80%D0%B5%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8)
* [6Проектирование программ в целом](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC_%D0%B2_%D1%86%D0%B5%D0%BB%D0%BE%D0%BC)
* [7Различные ООП-методологии](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#%D0%A0%D0%B0%D0%B7%D0%BB%D0%B8%D1%87%D0%BD%D1%8B%D0%B5_%D0%9E%D0%9E%D0%9F-%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B8)
  + [7.1Компонентное программирование](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#%D0%9A%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
  + [7.2Прототипное программирование](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D1%82%D0%B8%D0%BF%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
  + [7.3Класс-ориентированное программирование](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#%D0%9A%D0%BB%D0%B0%D1%81%D1%81-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
* [8Производительность объектных программ](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#%D0%9F%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D1%8B%D1%85_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC)
* [9Критика ООП](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#%D0%9A%D1%80%D0%B8%D1%82%D0%B8%D0%BA%D0%B0_%D0%9E%D0%9E%D0%9F)
* [10Объектно-ориентированные языки](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5_%D1%8F%D0%B7%D1%8B%D0%BA%D0%B8)
* [11См. также](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#%D0%A1%D0%BC._%D1%82%D0%B0%D0%BA%D0%B6%D0%B5)
* [12Примечания](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#%D0%9F%D1%80%D0%B8%D0%BC%D0%B5%D1%87%D0%B0%D0%BD%D0%B8%D1%8F)
* [13Литература](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#%D0%9B%D0%B8%D1%82%D0%B5%D1%80%D0%B0%D1%82%D1%83%D1%80%D0%B0)
* [14Ссылки](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B8)

## Основные понятия[[править](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&veaction=edit&section=1) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&section=1)]

[**Абстракция данных**](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%86%D0%B8%D1%8F_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)

Абстрагирование означает выделение значимой информации и исключение из рассмотрения незначимой. В ООП рассматривают лишь абстракцию данных (нередко называя её просто «абстракцией»), подразумевая набор наиболее значимых характеристик объекта, доступных остальной программе.

[**Инкапсуляция**](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))[[4]](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#cite_note-Pirs-4)

Инкапсуляция — свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе. Одни языки (например, [C++](https://ru.wikipedia.org/wiki/C%2B%2B), [Java](https://ru.wikipedia.org/wiki/Java" \o "Java) или [Ruby](https://ru.wikipedia.org/wiki/Ruby" \o "Ruby)) отождествляют инкапсуляцию с [сокрытием](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%BA%D1%80%D1%8B%D1%82%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), но другие ([Smalltalk](https://ru.wikipedia.org/wiki/Smalltalk" \o "Smalltalk), [Eiffel](https://ru.wikipedia.org/wiki/Eiffel" \o "Eiffel), [OCaml](https://ru.wikipedia.org/wiki/OCaml" \o "OCaml)) различают эти понятия.

[**Наследование**](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))[[4]](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#cite_note-Pirs-4)

Наследование — свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствованной функциональностью. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс — потомком, наследником, дочерним или производным классом.

[**Полиморфизм подтипов**](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_%D0%BF%D0%BE%D0%B4%D1%82%D0%B8%D0%BF%D0%BE%D0%B2)[[4]](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#cite_note-Pirs-4)

Полиморфизм подтипов (в ООП называемый просто «полиморфизмом») — свойство системы, позволяющее использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта. Другой вид полиморфизма — [параметрический](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BF%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC) — в ООП называют [обобщённым программированием](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

[**Класс**](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))

Класс — универсальный, комплексный [тип данных](https://ru.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), состоящий из тематически единого набора «полей» (переменных более элементарных типов) и «методов» (функций для работы с этими полями), то есть он является моделью информационной сущности с внутренним и внешним [интерфейсами](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_(%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) для оперирования своим содержимым (значениями полей). В частности, в классах широко используются специальные блоки из одного или чаще двух спаренных методов, отвечающих за элементарные операции с определённым полем (интерфейс присваивания и считывания значения, [геттер](https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D1%82%D1%82%D0%B5%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))-[сеттер](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D1%82%D0%B5%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))), которые имитируют непосредственный доступ к полю. Эти блоки называются «свойствами» и почти совпадают по конкретному имени со своим полем (например, имя поля может начинаться со строчной, а имя свойства — с заглавной буквы). Другим проявлением интерфейсной природы класса является то, что при копировании соответствующей переменной через присваивание копируется только интерфейс, но не сами данные, то есть класс — [ссылочный](https://ru.wikipedia.org/wiki/%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) тип данных. Переменная-объект, относящаяся к заданному классом типу, называется экземпляром этого класса. При этом в некоторых исполняющих системах класс также может представляться некоторым объектом при выполнении программы посредством [динамической идентификации типа данных](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%B8%D0%B4%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F_%D1%82%D0%B8%D0%BF%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Обычно классы разрабатывают таким образом, чтобы обеспечить отвечающие природе объекта и решаемой задаче целостность данных объекта, а также удобный и простой интерфейс. В свою очередь, целостность предметной области объектов и их интерфейсов, а также удобство их проектирования, обеспечивается наследованием.

[**Объект**](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))

Сущность в [адресном пространстве](https://ru.wikipedia.org/wiki/%D0%90%D0%B4%D1%80%D0%B5%D1%81%D0%B0%D1%86%D0%B8%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D0%B8) вычислительной системы, появляющаяся при создании экземпляра класса (например, после запуска результатов [компиляции](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) и [связывания](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D0%BD%D0%BA%D0%BE%D0%B2%D0%BA%D0%B0) исходного кода на выполнение).

## Классификация подвидов ООП[[править](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&veaction=edit&section=2) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&section=2)]

[Лука Карделли](https://ru.wikipedia.org/w/index.php?title=%D0%9A%D0%B0%D1%80%D0%B4%D0%B5%D0%BB%D0%BB%D0%B8,_%D0%9B%D1%83%D0%BA%D0%B0&action=edit&redlink=1)[[en]](https://en.wikipedia.org/wiki/Luca_Cardelli) и Мартин Абади построили теоретическое обоснование ООП и классификацию на основе этого обоснования[[5]](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#cite_note-_94fc08aa6ce66b8f-5)[[6]](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#cite_note-_141199d63e782555-6)[[7]](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#cite_note-_3979f653ce27c8be-7)[[8]](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#cite_note-_3cfdb020e79d8a32-8). Они отмечают, что выделенные ими понятия и категории вместе встречаются далеко не во всех ОО-языках, большинство языков поддерживают лишь подмножества теории, а порой и своеобразные отклонения от неё.

Ключевые понятия:

* [квантификация переменных типа](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B0#%D0%A1%D0%B2%D1%8F%D0%B7%D1%8B%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B8_%D0%BA%D0%B2%D0%B0%D0%BD%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F_%D0%BF%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D1%8B%D1%85_%D1%82%D0%B8%D0%BF%D0%B0) ([универсальная](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BF%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC), [экзистенциальная](https://ru.wikipedia.org/wiki/%D0%AD%D0%BA%D0%B7%D0%B8%D1%81%D1%82%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%82%D0%B8%D0%BF), [ограниченная](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B3%D1%80%D0%B0%D0%BD%D0%B8%D1%87%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BA%D0%B2%D0%B0%D0%BD%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F&action=edit&redlink=1)[[en]](https://en.wikipedia.org/wiki/Bounded_quantification));
* [подтипизация](https://ru.wikipedia.org/w/index.php?title=%D0%92%D1%8B%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D0%BE%D0%B4%D1%82%D0%B8%D0%BF%D0%BE%D0%B2_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85&action=edit&redlink=1)[[en]](https://en.wikipedia.org/wiki/Subtyping) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *subtyping* — отношения «супертип-подтип»);
* включение ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *subsumption* — см. [принцип подстановки Барбары Лисков](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF_%D0%BF%D0%BE%D0%B4%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BA%D0%B8_%D0%91%D0%B0%D1%80%D0%B1%D0%B0%D1%80%D1%8B_%D0%9B%D0%B8%D1%81%D0%BA%D0%BE%D0%B2)) — частный случай подтипизации — см. [подтипизация на записях](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)" \l "%D0%9F%D0%BE%D0%B4%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%BD%D0%B0_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D1%8F%D1%85" \o "Полиморфизм (информатика));
* [объект](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5));
* объектный тип (фактически, специальная форма «[записи](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BF%D0%B8%D1%81%D1%8C_(%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85))», в число полей которой входят [функции первого класса](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8_%D0%BF%D0%B5%D1%80%D0%B2%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B0));
* [класс](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)).

Класс описывает абстрактное поведение. Объектные типы строятся на основе класса посредством добавления различных приватных полей и методов. Объект (то есть значение объектного типа, традиционно называемое «экземпляром класса») порождается [конструктором](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D0%BE%D1%80_(%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) на основе начальных параметров.

Традиционно перечисляемые основные [принципы](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF)[Перейти к разделу «#Принципы»](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#%D0%9F%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF%D1%8B) ООП не апеллируют к данному теоретическому обоснованию, а являются устоявшимися в сообществе догмами (что приводит к сильному разбросу вариантов изложения в разных источниках). По большей части они относятся к языкам — потомкам [Алгола](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D0%BB) и [Симулы](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D1%83%D0%BB%D0%B0" \o "Симула); в меньшей степени — к потомкам [Smalltalk](https://ru.wikipedia.org/wiki/Smalltalk" \o "Smalltalk) (в частности, часто упоминаемый принцип [сокрытия](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%BA%D1%80%D1%8B%D1%82%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) в потомках [Smalltalk](https://ru.wikipedia.org/wiki/Smalltalk" \o "Smalltalk) семантически недоступен и идеологически считается несущественным). В значительно большей степени теоретически обоснованные концепции ООП поддерживают ОО-языки, развившиеся на поприще [функционального программирования](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5): [OCaml](https://ru.wikipedia.org/wiki/OCaml" \o "OCaml), диалекты [Haskell](https://ru.wikipedia.org/wiki/Haskell" \o "Haskell) (O’Haskell, Mondrian), [successor ML](https://ru.wikipedia.org/w/index.php?title=Successor_ML&action=edit&redlink=1" \o "Successor ML (страница отсутствует)). Более того, основные идеи объектного моделирования в этом случае не требуют прямой поддержки со стороны языка, а могут быть сравнительно легко эмулированы[[9]](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#cite_note-9).

Наиболее заметные отличия в проявлении [показателей качества](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%87%D0%B5%D1%81%D1%82%D0%B2%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) между языками разных видов:

* В [мейнстримных](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B9%D0%BD%D1%81%D1%82%D1%80%D0%B8%D0%BC" \o "Мейнстрим) языках декларируемые принципы нацелены на повышение изначально низкого для [императивного программирования](https://ru.wikipedia.org/wiki/%D0%98%D0%BC%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) коэффициента [повторного использования кода](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B2%D1%82%D0%BE%D1%80%D0%BD%D0%BE%D0%B5_%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B0). В [полиморфно типизированных](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_%D1%82%D0%B8%D0%BF%D0%BE%D0%B2) применение концепций ООП, напротив, означает очевидное его снижение из-за перехода от [параметрического полиморфизма](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BF%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC) к [ad-hoc-полиморфизму](https://ru.wikipedia.org/wiki/Ad-hoc-%D0%BF%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC" \o "Ad-hoc-полиморфизм)[[10]](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#cite_note-_63598c1968cd3197-10). В [динамически типизированных](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) языках ([Smalltalk](https://ru.wikipedia.org/wiki/Smalltalk" \o "Smalltalk), [Python](https://ru.wikipedia.org/wiki/Python" \o "Python), [Ruby](https://ru.wikipedia.org/wiki/Ruby" \o "Ruby)) эти принципы используются для логической организации программы, и их влияние на коэффициент [повторного использования](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B2%D1%82%D0%BE%D1%80%D0%BD%D0%BE%D0%B5_%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B0) трудно спрогнозировать — он сильно зависит от дисциплины программиста. Например, в [CLOS](https://ru.wikipedia.org/wiki/CLOS) [мультиметоды](https://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BC%D0%B5%D1%82%D0%BE%D0%B4" \o "Мультиметод) одновременно являются [функциями первого класса](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8_%D0%BF%D0%B5%D1%80%D0%B2%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B0), что позволяет рассматривать их одновременно и как [связанно квантифицированные](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B2%D1%8F%D0%B7%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BA%D0%B2%D0%B0%D0%BD%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F&action=edit&redlink=1)[[en]](https://en.wikipedia.org/wiki/Bounded_quantification), и как [обобщённые](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) ([истинно полиморфные](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BF%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC)).
* Традиционные ОО-языки используют [номинативную типизацию](https://ru.wikipedia.org/w/index.php?title=%D0%9D%D0%BE%D0%BC%D0%B8%D0%BD%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F&action=edit&redlink=1)[[en]](https://en.wikipedia.org/wiki/Nominal_type_system), то есть допустимость соиспользования объектов разных классов только при условии явного указания родственных отношений между классами. Для [полиморфно типизированных](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_%D1%82%D0%B8%D0%BF%D0%BE%D0%B2) языков характерна [структурная типизация](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F&action=edit&redlink=1)[[en]](https://en.wikipedia.org/wiki/Structural_type_system), то есть согласование классов между собой тем же механизмом, что и согласование числа 5 с типом [int](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BB%D1%8B%D0%B9_%D1%82%D0%B8%D0%BF" \o "Целый тип). [Динамически типизированные](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) языки также занимают здесь промежуточную позицию.

Обобщённое обоснование [динамической диспетчеризации](https://ru.wikipedia.org/w/index.php?title=%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%B4%D0%B8%D1%81%D0%BF%D0%B5%D1%82%D1%87%D0%B5%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F&action=edit&redlink=1)[[en]](https://en.wikipedia.org/wiki/Dynamic_dispatch) (включая [множественную](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B4%D0%B8%D1%81%D0%BF%D0%B5%D1%82%D1%87%D0%B5%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F)) в середине [1990-х годов](https://ru.wikipedia.org/wiki/1990-%D0%B5_%D0%B3%D0%BE%D0%B4%D1%8B) построил Джузеппе Кастанья[[11]](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5" \l "cite_note-11)[[12]](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#cite_note-12)[[13]](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#cite_note-_70d489692de7d5ec-13).

## История[[править](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&veaction=edit&section=3) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&section=3)]

|  |  |
| --- | --- |
| [https://upload.wikimedia.org/wikipedia/commons/thumb/0/09/Nuvola_apps_important_recycle.svg/45px-Nuvola_apps_important_recycle.svg.png](https://commons.wikimedia.org/wiki/File:Nuvola_apps_important_recycle.svg?uselang=ru) | **Эта статья или раздел нуждается в переработке.**  Пожалуйста, улучшите статью в соответствии с [правилами написания статей](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F:%D0%9F%D1%80%D0%B0%D0%B2%D0%B8%D0%BB%D0%B0_%D0%B8_%D1%83%D0%BA%D0%B0%D0%B7%D0%B0%D0%BD%D0%B8%D1%8F). |

**Объе́ктно-ориенти́рованное программирование** возникло в результате развития идеологии [процедурного программирования](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), где данные и подпрограммы (процедуры, функции) их обработки формально не связаны. Для дальнейшего развития объектно-ориентированного программирования часто большое значение имеют понятия события (так называемое [событийно-ориентированное программирование](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) и компонента ([компонентное программирование](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), КОП).

Взаимодействие объектов происходит посредством [сообщений](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#%D0%A1%D0%BE%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%B8%D0%B5). Результатом дальнейшего развития ООП, по-видимому, будет [агентно-ориентированое программирование](https://ru.wikipedia.org/wiki/%D0%90%D0%B3%D0%B5%D0%BD%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BF%D0%BE%D0%B4%D1%85%D0%BE%D0%B4" \o "Агентно-ориентированный подход), где *агенты* — независимые части кода на уровне выполнения. Взаимодействие агентов происходит посредством изменения *среды*, в которой они находятся.

Языковые конструкции, конструктивно не относящиеся непосредственно к объектам, но сопутствующие им для их безопасной ([исключительные ситуации](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B9), проверки) и эффективной работы, инкапсулируются от них в аспекты (в [аспектно-ориентированном программировании](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D0%BF%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5" \o "Аспектно-ориентированное программирование)). [Субъектно-ориентированное программирование](https://ru.wikipedia.org/wiki/%D0%A1%D1%83%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) расширяет понятие объекта посредством обеспечения более унифицированного и независимого взаимодействия объектов. Может являться переходной стадией между ООП и агентным программированием в части самостоятельного их взаимодействия.

Первым языком программирования, в котором были предложены основные понятия, впоследствии сложившиеся в парадигму, была [Симула](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D1%83%D0%BB%D0%B0" \o "Симула), но термин «объектная ориентированность» не использовался в контексте использования этого языка. В момент его появления в [1967 году](https://ru.wikipedia.org/wiki/1967_%D0%B3%D0%BE%D0%B4) в нём были предложены революционные идеи: объекты, классы, [виртуальные методы](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4) и др., однако это всё не было воспринято современниками как нечто грандиозное. Фактически, Симула была «Алголом с классами», упрощающим выражение в [процедурном программировании](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) многих сложных концепций. Понятие класса в Симуле может быть полностью определено через композицию конструкций Алгола (то есть класс в Симуле — это нечто сложное, описываемое посредством примитивов).

Взгляд на программирование «под новым углом» (отличным от процедурного) предложили [Алан Кэй](https://ru.wikipedia.org/wiki/%D0%9A%D1%8D%D0%B9,_%D0%90%D0%BB%D0%B0%D0%BD_%D0%9A%D1%83%D1%80%D1%82%D0%B8%D1%81) и [Дэн Ингаллс](https://ru.wikipedia.org/w/index.php?title=%D0%98%D0%BD%D0%B3%D0%B0%D0%BB%D0%BB%D1%81,_%D0%94%D1%8D%D0%BD&action=edit&redlink=1) в языке [Smalltalk](https://ru.wikipedia.org/wiki/Smalltalk" \o "Smalltalk). Здесь понятие класса стало основообразующей идеей для всех остальных конструкций языка (то есть класс в Смолтоке является примитивом, посредством которого описаны более сложные конструкции). Именно он стал первым широко распространённым [объектно-ориентированным языком программирования](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F).

В настоящее время количество прикладных языков программирования ([список языков](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F#%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D1%8F%D0%B7%D1%8B%D0%BA%D0%BE%D0%B2)), реализующих объектно-ориентированную парадигму, является наибольшим по отношению к другим парадигмам. Наиболее распространённые в промышленности языки (C++, Delphi, C#, Java и др.) воплощают объектную модель Симулы. Примерами языков, опирающихся на модель Смолтока, являются Objective-C, Python, Ruby.

## Определение ООП и его основные концепции[[править](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&veaction=edit&section=4) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&section=4)]

В центре ООП находится понятие *объекта.* Объект — это сущность, которой можно посылать сообщения и которая может на них реагировать, используя свои данные. Объект — это экземпляр класса. Данные объекта скрыты от остальной программы. [Инкапсуляция](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) включает в себя [сокрытие](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%BA%D1%80%D1%8B%D1%82%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) (Но им не является!).

Наличие инкапсуляции достаточно для объектности языка программирования, но ещё не означает его объектной ориентированности — для этого требуется наличие [наследования](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)).

Но даже наличие инкапсуляции и наследования не делает язык программирования в полной мере объектным с точки зрения ООП. Основные преимущества ООП проявляются только в том случае, когда в языке программирования реализован [полиморфизм подтипов](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_%D0%BF%D0%BE%D0%B4%D1%82%D0%B8%D0%BF%D0%BE%D0%B2) — возможность единообразно обрабатывать объекты с различной реализацией при условии наличия общего интерфейса.

### Сложности определения[[править](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&veaction=edit&section=5) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&section=5)]

|  |  |
| --- | --- |
| [https://upload.wikimedia.org/wikipedia/commons/thumb/2/2c/Broom_icon.svg/40px-Broom_icon.svg.png](https://commons.wikimedia.org/wiki/File:Broom_icon.svg?uselang=ru) | В этой статье имеется **избыток цитат либо слишком длинные цитаты**.  [Излишние и чрезмерно большие цитаты](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F:%D0%A6%D0%B8%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) следует обобщить и переписать своими словами. Возможно, эти цитаты будут более уместны в [Викицитатнике](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D1%86%D0%B8%D1%82%D0%B0%D1%82%D0%BD%D0%B8%D0%BA) или в [Викитеке](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D1%82%D0%B5%D0%BA%D0%B0). |

ООП имеет уже более чем сорокалетнюю историю, но, несмотря на это, до сих пор не существует чёткого общепринятого определения данной технологии[[14]](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#cite_note-14). Основные принципы, заложенные в первые объектные языки и системы, подверглись существенному изменению (или искажению) и дополнению при многочисленных реализациях последующего времени. Кроме того, примерно с середины [1980-х годов](https://ru.wikipedia.org/wiki/1980-%D0%B5_%D0%B3%D0%BE%D0%B4%D1%8B) термин «объектно-ориентированный» стал [модным](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B0), в результате с ним произошло то же самое, что несколько раньше с термином «структурный» (ставшим модным после распространения технологии [структурного программирования](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) — его стали искусственно «прикреплять» к любым новым разработкам, чтобы обеспечить им привлекательность. [Бьёрн Страуструп](https://ru.wikipedia.org/wiki/%D0%91%D1%8C%D1%91%D1%80%D0%BD_%D0%A1%D1%82%D1%80%D0%B0%D1%83%D1%81%D1%82%D1%80%D1%83%D0%BF" \o "Бьёрн Страуструп) в [1988 году](https://ru.wikipedia.org/wiki/1988_%D0%B3%D0%BE%D0%B4) писал, что обоснование «объектной ориентированности» чего-либо, в большинстве случаев, сводится к некорректному [силлогизму](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BB%D0%BB%D0%BE%D0%B3%D0%B8%D0%B7%D0%BC): «X — это хорошо. Объектная ориентированность — это хорошо. *Следовательно*, X является объектно-ориентированным».

Тимоти Бадд пишет[[15]](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#cite_note-badd-15)[[16]](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#cite_note-16):

Роджер Кинг аргументированно настаивал, что его кот является объектно-ориентированным. Кроме прочих своих достоинств, кот демонстрирует характерное поведение, реагирует на сообщения, наделён унаследованными реакциями и управляет своим, вполне независимым, внутренним состоянием.

По мнению [Алана Кэя](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B0%D0%BD_%D0%9A%D1%8D%D0%B9), создателя языка [Smalltalk](https://ru.wikipedia.org/wiki/Smalltalk" \o "Smalltalk), которого считают одним из «отцов-основателей» ООП, объектно-ориентированный подход заключается в следующем наборе основных принципов (цитируется по вышеупомянутой книге Т. Бадда).

1. Всё является объектом.
2. Вычисления осуществляются путём взаимодействия (обмена данными) между объектами, при котором один объект требует, чтобы другой объект выполнил некоторое действие. Объекты взаимодействуют, посылая и получая сообщения. Сообщение — это запрос на выполнение действия, дополненный набором аргументов, которые могут понадобиться при выполнении действия.
3. Каждый объект имеет независимую память, которая состоит из других объектов.
4. Каждый объект является представителем класса, который выражает общие свойства объектов (таких, как целые числа или списки).
5. В классе задаётся поведение (функциональность) объекта. Тем самым все объекты, которые являются экземплярами одного класса, могут выполнять одни и те же действия.
6. Классы организованы в единую древовидную структуру с общим корнем, называемую иерархией наследования. Память и поведение, связанное с экземплярами определённого класса, автоматически доступны любому классу, расположенному ниже в иерархическом дереве.

Таким образом, программа представляет собой набор объектов, имеющих состояние и поведение. Объекты взаимодействуют посредством сообщений. Естественным образом выстраивается иерархия объектов: программа в целом — это объект, для выполнения своих функций она обращается к входящим в неё объектам, которые, в свою очередь, выполняют запрошенное путём обращения к другим объектам программы. Естественно, чтобы избежать [бесконечной рекурсии](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%81%D0%BA%D0%BE%D0%BD%D0%B5%D1%87%D0%BD%D0%B0%D1%8F_%D1%80%D0%B5%D0%BA%D1%83%D1%80%D1%81%D0%B8%D1%8F) в обращениях, на каком-то этапе объект трансформирует обращённое к нему сообщение в сообщения к стандартным системным объектам, предоставляемым языком и средой программирования.

Устойчивость и управляемость системы обеспечивается за счёт чёткого разделения ответственности объектов (за каждое действие отвечает определённый объект), однозначного определения интерфейсов межобъектного взаимодействия и полной изолированности внутренней структуры объекта от внешней среды (инкапсуляции).

Определить ООП можно и многими другими способами.

### Концепции[[править](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&veaction=edit&section=6) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&section=6)]

Появление в ООП отдельного понятия **класса** закономерно вытекает из желания иметь множество объектов со сходным поведением. Класс в ООП — это в чистом виде [абстрактный тип данных](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BD%D1%8B%D0%B9_%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), создаваемый программистом. С этой точки зрения объекты являются значениями данного абстрактного типа, а определение класса задаёт внутреннюю структуру значений и набор операций, которые над этими значениями могут быть выполнены. Желательность иерархии классов (а значит, наследования) вытекает из требований к повторному использованию кода — если несколько классов имеют сходное поведение, нет смысла дублировать их описание, лучше выделить общую часть в общий родительский класс, а в описании самих этих классов оставить только различающиеся элементы.

Необходимость совместного использования объектов разных классов, способных обрабатывать однотипные сообщения, требует поддержки [**полиморфизма**](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) — возможности записывать разные объекты в переменные одного и того же типа. В таких условиях объект, отправляя сообщение, может не знать в точности, к какому классу относится адресат, и одни и те же сообщения, отправленные переменным одного типа, содержащим объекты разных классов, вызовут различную реакцию.

Отдельного пояснения требует понятие обмена [**сообщениями**](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%B8%D0%B5). Первоначально (например, в том же [Smalltalk](https://ru.wikipedia.org/wiki/Smalltalk" \o "Smalltalk)) взаимодействие объектов представлялось как «настоящий» обмен сообщениями, то есть пересылка от одного объекта другому специального объекта-сообщения. Такая модель является чрезвычайно общей. Она прекрасно подходит, например, для описания параллельных вычислений с помощью *активных объектов*, каждый из которых имеет собственный поток исполнения и работает одновременно с прочими. Такие объекты могут вести себя как отдельные, абсолютно автономные вычислительные единицы. Посылка сообщений естественным образом решает вопрос обработки сообщений объектами, присвоенными полиморфным переменным — независимо от того, как объявляется переменная, сообщение обрабатывает код класса, к которому относится присвоенный переменной объект. Данный подход реализован в языках программирования [Smalltalk](https://ru.wikipedia.org/wiki/Smalltalk" \o "Smalltalk), [Ruby](https://ru.wikipedia.org/wiki/Ruby" \o "Ruby), [Objective-C](https://ru.wikipedia.org/wiki/Objective-C" \o "Objective-C), [Python](https://ru.wikipedia.org/wiki/Python" \o "Python).

Однако общность механизма обмена сообщениями имеет и другую сторону — «полноценная» передача сообщений требует дополнительных накладных расходов, что не всегда приемлемо. Поэтому во многих современных объектно-ориентированных языках программирования используется концепция **«отправка сообщения как вызов метода»** — объекты имеют доступные извне методы, вызовами которых и обеспечивается взаимодействие объектов. Данный подход реализован в огромном количестве языков программирования, в том числе [C++](https://ru.wikipedia.org/wiki/C%2B%2B), [Object Pascal](https://ru.wikipedia.org/wiki/Object_Pascal" \o "Object Pascal), [Java](https://ru.wikipedia.org/wiki/Java" \o "Java), [Oberon-2](https://ru.wikipedia.org/wiki/Oberon-2). Однако, это приводит к тому, что сообщения уже не являются самостоятельными объектами, и, как следствие, не имеют атрибутов, что сужает возможности программирования. Некоторые языки используют гибридное представление, демонстрируя преимущества одновременно обоих подходов — например, [CLOS](https://ru.wikipedia.org/wiki/CLOS), [Python](https://ru.wikipedia.org/wiki/Python" \o "Python).

Концепция **виртуальных методов**, поддерживаемая этими и другими современными языками, появилась как средство обеспечить выполнение нужных методов при использовании полиморфных переменных, то есть, по сути, как попытка расширить возможности вызова методов для реализации части функциональности, обеспечиваемой механизмом обработки сообщений.

## Особенности реализации[[править](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&veaction=edit&section=7) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&section=7)]

Как уже говорилось выше, в современных объектно-ориентированных языках программирования каждый объект является значением, относящимся к определённому [классу](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). Класс представляет собой объявленный программистом составной [тип данных](https://ru.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), имеющий в составе:

[**Поля данных**](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B5_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B0)

Параметры объекта (конечно, не все, а только необходимые в программе), задающие его состояние (свойства объекта предметной области). Иногда поля данных объекта называют свойствами объекта, из-за чего возможна путаница. Фактически поля представляют собой значения (переменные, константы), объявленные как принадлежащие классу.

[**Методы**](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_(%D1%8F%D0%B7%D1%8B%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F))

Процедуры и функции, связанные с классом. Они определяют действия, которые можно выполнять над объектом такого типа, и которые сам объект может выполнять.

Классы могут наследоваться друг от друга. Класс-потомок получает все поля и методы класса-родителя, но может дополнять их собственными либо переопределять уже имеющиеся. Большинство языков программирования поддерживает только единичное наследование (класс может иметь только один класс-родитель), лишь в некоторых допускается множественное наследование — порождение класса от двух или более классов-родителей. Множественное наследование создаёт целый ряд проблем, как логических, так и чисто реализационных, поэтому в полном объёме его поддержка не распространена. Вместо этого в 1990-е годы появилось и стало активно вводиться в объектно-ориентированные языки понятие [интерфейса](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_(%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). Интерфейс — это класс без полей и без реализации, включающий только заголовки методов. Если некий класс наследует (или, как говорят, реализует) интерфейс, он должен реализовать все входящие в него методы. Использование интерфейсов предоставляет относительно дешёвую альтернативу множественному наследованию.

Взаимодействие объектов в абсолютном большинстве случаев обеспечивается вызовом ими методов друг друга.

Инкапсуляция обеспечивается следующими средствами:

**Контроль доступа**

Поскольку методы класса могут быть как чисто внутренними, обеспечивающими логику функционирования объекта, так и внешними, с помощью которых взаимодействуют объекты, необходимо обеспечить скрытость первых при доступности извне вторых. Для этого в языки вводятся специальные синтаксические конструкции, явно задающие область видимости каждого члена класса. Традиционно это модификаторы public, protected и private, обозначающие, соответственно, открытые члены класса, члены класса, доступные внутри класса и из классов-потомков, и скрытые, доступные только внутри класса. Конкретная номенклатура модификаторов и их точный смысл различаются в разных языках.

**Методы доступа**

Поля класса в общем случае не должны быть доступны извне, поскольку такой доступ позволил бы произвольным образом менять внутреннее состояние объектов. Поэтому поля обычно объявляются скрытыми (либо язык в принципе не позволяет обращаться к полям класса извне), а для доступа к находящимся в полях данным используются специальные методы, называемые методами доступа. Такие методы либо возвращают значение того или иного поля, либо производят запись в это поле нового значения. При записи метод доступа может проконтролировать допустимость записываемого значения и, при необходимости, произвести другие манипуляции с данными объекта, чтобы они остались корректными (внутренне согласованными). Методы доступа называют ещё аксессорами (от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *access* — доступ), а по отдельности — [геттерами](https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D1%82%D1%82%D0%B5%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *get* — получить) и [сеттерами](https://ru.wikipedia.org/wiki/Setter) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *set* — установить)[[17]](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#cite_note-17).

[**Свойства объекта**](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))

Псевдополя, доступные для чтения и/или записи. Свойства внешне выглядят как поля и используются аналогично доступным полям (с некоторыми исключениями), однако фактически при обращении к ним происходит вызов методов доступа. Таким образом, свойства можно рассматривать как «умные» поля данных, сопровождающие доступ к внутренним данным объекта какими-либо дополнительными действиями (например, когда изменение координаты объекта сопровождается его перерисовкой на новом месте). Свойства, по сути, не более чем [**синтаксический сахар**](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%81%D0%B0%D1%85%D0%B0%D1%80), поскольку никаких новых возможностей они не добавляют, а лишь скрывают вызов методов доступа. Конкретная языковая реализация свойств может быть разной. Например, в [C#](https://ru.wikipedia.org/wiki/C_Sharp) объявление свойства непосредственно содержит код методов доступа, который вызывается только при работе со свойствами, то есть не требует отдельных методов доступа, доступных для непосредственного вызова. В Delphi объявление свойства содержит лишь имена методов доступа, которые должны вызываться при обращении к полю. Сами методы доступа представляют собой обычные методы с некоторыми дополнительными требованиями к [сигнатуре](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%B3%D0%BD%D0%B0%D1%82%D1%83%D1%80%D0%B0_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8).

Полиморфизм реализуется путём введения в язык правил, согласно которым переменной типа «класс» может быть присвоен объект любого класса-потомка её класса.

## Проектирование программ в целом[[править](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&veaction=edit&section=8) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&section=8)]

ООП ориентировано на разработку крупных программных комплексов, разрабатываемых командой программистов (возможно, достаточно большой). Проектирование системы в целом, создание отдельных компонентов и их объединение в конечный продукт при этом часто выполняется разными людьми, и нет ни одного специалиста, который знал бы о проекте всё.

Объектно-ориентированное проектирование ориентируется на описание структуры проектируемой системы (приоритетно по отношению к описанию её поведения, в отличие от [функционального программирования](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), то есть, фактически, в ответе на два основных вопроса:

* **Из каких частей состоит система**;
* **В чём состоит ответственность каждой из её частей**.

Выделение частей производится таким образом, чтобы каждая имела минимальный по объёму и точно определённый набор выполняемых функций (обязанностей), и при этом взаимодействовала с другими частями как можно меньше.

Дальнейшее уточнение приводит к выделению более мелких фрагментов описания. По мере детализации описания и определения ответственности выявляются данные, которые необходимо хранить, наличие близких по поведению агентов, которые становятся кандидатами на реализацию в виде классов с общими предками. После выделения компонентов и определения интерфейсов между ними реализация каждого компонента может проводиться практически независимо от остальных (разумеется, при соблюдении соответствующей технологической дисциплины).

Большое значение имеет правильное построение иерархии классов. Одна из известных проблем больших систем, построенных по ООП-технологии — так называемая *проблема хрупкости базового класса*. Она состоит в том, что на поздних этапах разработки, когда иерархия классов построена и на её основе разработано большое количество кода, оказывается трудно или даже невозможно внести какие-либо изменения в код базовых классов иерархии (от которых порождены все или многие работающие в системе классы). Даже если вносимые изменения не затронут интерфейс базового класса, изменение его поведения может непредсказуемым образом отразиться на классах-потомках. В случае крупной системы разработчик базового класса просто не в состоянии предугадать последствия изменений, он даже не знает о том, как именно базовый класс используется и от каких особенностей его поведения зависит корректность работы классов-потомков.

## Различные ООП-методологии[[править](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&veaction=edit&section=9) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&section=9)]

Компонентное программирование — следующий этап развития ООП; прототип- и класс-ориентированное программирование — разные подходы к созданию программы, которые могут комбинироваться, имеющие свои преимущества и недостатки.

### Компонентное программирование[[править](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&veaction=edit&section=10) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&section=10)]

*Основная статья:*[***Компонентно-ориентированное программирование***](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)

[Компонентно-ориентированное программирование](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) — это своеобразная «надстройка» над ООП, набор правил и ограничений, направленных на построение крупных развивающихся программных систем с большим временем жизни. Программная система в этой методологии представляет собой набор компонентов с хорошо определёнными интерфейсами. Изменения в существующую систему вносятся путём создания новых компонентов в дополнение или в качестве замены ранее существующих. При создании новых компонентов на основе ранее созданных запрещено использование наследования реализации — новый компонент может наследовать лишь интерфейсы базового. Таким образом, компонентное программирование обходит проблему хрупкости базового класса.

### Прототипное программирование[[править](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&veaction=edit&section=11) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&section=11)]

[Прототипное программирование](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D1%82%D0%B8%D0%BF%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), сохранив часть черт ООП, отказалось от базовых понятий — класса и наследования.

* Прототип — это объект-образец, по образу и подобию которого создаются другие объекты. Объекты-копии могут сохранять связь с родительским объектом, автоматически наследуя изменения в прототипе; эта особенность определяется в рамках конкретного [языка](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F).
* Вместо механизма описания классов и порождения экземпляров, язык предоставляет механизм создания объекта (путём задания набора полей и методов, которые объект должен иметь) и механизм клонирования объектов.
* Каждый вновь созданный объект является «экземпляром без класса». Каждый объект может стать *прототипом* — быть использован для создания нового объекта с помощью операции *клонирования*. После клонирования новый объект может быть изменён, в частности, дополнен новыми полями и методами.
* Клонированный объект либо становится полной копией прототипа, хранящей все значения его полей и дублирующей его методы, либо сохраняет ссылку на прототип, не включая в себя клонированных полей и методов до тех пор, пока они не будут изменены. В последнем случае среда исполнения обеспечивает механизм *делегирования* — если при обращении к объекту он сам не содержит нужного метода или поля данных, вызов передаётся прототипу, от него, при необходимости — дальше по цепочке.

### Класс-ориентированное программирование[[править](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&veaction=edit&section=12) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&section=12)]

|  |  |
| --- | --- |
| https://upload.wikimedia.org/wikipedia/commons/thumb/6/64/Question_book-4.svg/48px-Question_book-4.svg.png | В разделе **не хватает**[**ссылок на источники**](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F:%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B8_%D0%BD%D0%B0_%D0%B8%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8) (см. [рекомендации по поиску](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F:%D0%9F%D0%BE%D0%B8%D1%81%D0%BA_%D0%B8%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%BE%D0%B2" \o "Википедия:Поиск источников)).  Информация должна быть [проверяема](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F:%D0%9F%D1%80%D0%BE%D0%B2%D0%B5%D1%80%D1%8F%D0%B5%D0%BC%D0%BE%D1%81%D1%82%D1%8C), иначе она может быть удалена. Вы можете [отредактировать](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit) статью, добавив ссылки на [авторитетные источники](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F:%D0%90%D0%B2%D1%82%D0%BE%D1%80%D0%B8%D1%82%D0%B5%D1%82%D0%BD%D1%8B%D0%B5_%D0%B8%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8) в виде [сносок](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F:%D0%A1%D0%BD%D0%BE%D1%81%D0%BA%D0%B8). *(31 декабря 2012)* |

Класс-ориентированное программирование — это программирование, сфокусированное на данных, причём данные и поведение неразрывно связаны между собой. Вместе данные и поведение представляют собой класс. Соответственно в языках, основанных на понятии «класс», все объекты разделены на два основных типа — классы и экземпляры. Класс определяет структуру и функциональность (поведение), одинаковую для всех экземпляров данного класса. Экземпляр является носителем данных — то есть обладает состоянием, меняющимся в соответствии с поведением, заданным классом. В класс-ориентированных языках новый экземпляр создаётся через вызов конструктора класса (возможно, с набором параметров). Получившийся экземпляр имеет структуру и поведение, жёстко заданные его классом.

## Производительность объектных программ[[править](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&veaction=edit&section=13) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&section=13)]

[Гради Буч](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%B4%D0%B8_%D0%91%D1%83%D1%87) указывает[[18]](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5#cite_note-booch-18) на следующие причины, приводящие к снижению производительности программ из-за использования объектно-ориентированных средств:

**Динамическое связывание методов**

Обеспечение полиморфного поведения объектов приводит к необходимости связывать методы, вызываемые программой (то есть определять, какой конкретно метод будет вызываться) не на этапе компиляции, а в процессе исполнения программы, на что тратится дополнительное время. При этом реально динамическое связывание требуется не более чем для 20 % вызовов, но некоторые ООП-языки используют его постоянно.

**Значительная глубина абстракции**

ООП-разработка часто приводит к созданию «многослойных» приложений, где выполнение объектом требуемого действия сводится к множеству обращений к объектам более низкого уровня. В таком приложении происходит очень много вызовов методов и возвратов из методов, что, естественно, сказывается на производительности.

**Наследование «размывает» код**

Код, относящийся к «конечным» классам иерархии наследования, которые обычно и используются программой непосредственно, находится не только в самих этих классах, но и в их классах-предках. Относящиеся к одному классу методы фактически описываются в разных классах. Это приводит к двум неприятным моментам:

* **Снижается скорость трансляции, так как компоновщику приходится подгружать описания всех классов иерархии.**
* **Снижается производительность программы в системе со страничной памятью — поскольку методы одного класса физически находятся в разных местах кода, далеко друг от друга, при работе фрагментов программы, активно обращающихся к унаследованным методам, система вынуждена производить частые переключения страниц.**

**Инкапсуляция снижает скорость доступа к данным**

Запрет на прямой доступ к полям класса извне приводит к необходимости создания и использования методов доступа. И написание, и компиляция, и исполнение методов доступа сопряжены с дополнительными расходами.

**Динамическое создание и уничтожение объектов**

Динамически создаваемые объекты, как правило, размещаются в [куче](https://ru.wikipedia.org/wiki/%D0%9A%D1%83%D1%87%D0%B0_(%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C)), что менее эффективно, чем размещение их на стеке и, тем более, статическое выделение памяти под них на этапе компиляции.