**Гибкая методология разработки** ([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) *Agile software development*, *agile-методы*) — серия подходов к [разработке программного обеспечения](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%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), ориентированных на использование итеративной разработки, динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля[[1]](https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%B1%D0%BA%D0%B0%D1%8F_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8#cite_note-1).

Существует несколько методик, относящихся к классу гибких методологий разработки, в частности [экстремальное программирование](https://ru.wikipedia.org/wiki/%D0%AD%D0%BA%D1%81%D1%82%D1%80%D0%B5%D0%BC%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), [Scrum](https://ru.wikipedia.org/wiki/Scrum), [FDD](https://ru.wikipedia.org/wiki/Feature_driven_development).

**Feature driven development** (**FDD**, *разработка, управляемая функциональностью*). Основной целью данной методологии является разработка реального, работающего программного обеспечения систематически, в поставленные сроки. FDD включает в себя пять базовых видов деятельности:

1. разработка общей модели;
2. составление списка необходимых функций системы;
3. планирование работы над каждой функцией;
4. проектирование функции;

пишутся «болванки» классов и методов, и происходит критическое рассмотрение дизайна.

1. реализация функции

программный код + тесты

FDD построен на основе набора передового опыта (набора наилучших практик), признанного в отрасли и полученного из [инженерии программного обеспечения](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D0%B8%D1%8F_%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). Эти практические методы строятся с точки зрения важного для клиента функционала. Ниже дано краткое описание каждого метода:

* *Объектное моделирование области.* Объектное моделирование состоит из исследования и выяснения рамок предметной области решаемой задачи. Результатом является общий каркас, который можно в дальнейшем дополнять функциями.
* *Разработка по функции.* Любая функция, которая слишком сложна для разработки в течение двух недель, разбивается на меньшие подфункции до тех пор, пока каждая подзадача не может быть названа свойством (то есть, быть реализована за 2 недели). Это облегчает создание корректно работающих функций, расширение и модификацию системы.
* *Индивидуальное владение классом (кодом).* Означает, что каждый блок кода закреплён за конкретным владельцем-разработчиком. Владелец ответственен за согласованность, производительность и концептуальную целостность своих классов.
* *Команда по разработке функций (свойств)*. Команда по разработке функций (свойств) — маленькая, динамически формируемая команда разработчиков, занимающаяся небольшой подзадачей. Позволяет нескольким разработчикам участвовать в дизайне свойства, а также оценивать дизайнерские решения перед выбором наилучшего.
* [*Проверка кода*](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%81%D0%BC%D0%BE%D1%82%D1%80_%D0%BA%D0%BE%D0%B4%D0%B0) ([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) *code review*) Проверки обеспечивают хорошее качество кода, в первую очередь путём выявления ошибок.
* [*Конфигурационное управление*](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%84%D0%B8%D0%B3%D1%83%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5). Помогает с идентификацией исходного кода для всех функций (свойств), разработка которых завершена на текущий момент, и с протоколированием изменений, сделанных разработчиками классов.
* *Регулярная сборка.* Регулярная сборка гарантирует, что всегда есть продукт (система), которая может быть представлена заказчику, и помогает находить ошибки при объединении частей исходного кода на ранних этапах.
* *Обозримость хода работ и результатов.* Частые и точные отчёты о ходе выполнения работ на всех уровнях внутри и за пределами проекта о выполненной работе помогают менеджерам правильно руководить проектом.

**Экстрема́льное программи́рование** ([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) *Extreme Programming*, *XP*) — одна из [гибких методологий разработки](https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%B1%D0%BA%D0%B0%D1%8F_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) [программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5). Авторы методологии — [Кент Бек](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D0%BA,_%D0%9A%D0%B5%D0%BD%D1%82), [Уорд Каннингем](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BD%D0%BD%D0%B8%D0%BD%D0%B3%D0%B5%D0%BC,_%D0%A3%D0%BE%D1%80%D0%B4), [Мартин Фаулер](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%80%D1%82%D0%B8%D0%BD_%D0%A4%D0%B0%D1%83%D0%BB%D0%B5%D1%80) и другие.

Двенадцать основных приёмов экстремального программирования (по первому изданию книги *Extreme programming explained*) могут быть объединены в четыре группы:

* Короткий цикл обратной связи (Fine-scale feedback)
  + [Разработка через тестирование](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D1%87%D0%B5%D1%80%D0%B5%D0%B7_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) (Test-driven development)
  + Игра в планирование (Planning game)
  + Заказчик всегда рядом (Whole team, Onsite customer)
  + [Парное программирование](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%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) (Pair programming)
* Непрерывный, а не пакетный процесс
  + [Непрерывная интеграция](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BF%D1%80%D0%B5%D1%80%D1%8B%D0%B2%D0%BD%D0%B0%D1%8F_%D0%B8%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D1%86%D0%B8%D1%8F) (Continuous integration)
  + [Рефакторинг](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3) (Design improvement, Refactoring)
  + Частые небольшие релизы (Small releases)
* Понимание, разделяемое всеми
  + Простота (Simple design)
  + Метафора системы (System metaphor)
  + Коллективное владение кодом (Collective code ownership) или выбранными шаблонами проектирования (Collective patterns ownership)
  + [Стандарт кодирования](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82_%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) (Coding standard or Coding conventions)
* Социальная защищённость программиста (Programmer welfare):
  + 40-часовая рабочая неделя (Sustainable pace, Forty-hour week)

Особое внимание уделяется двум разновидностям тестирования:

* [юнит-тестирование](https://ru.wikipedia.org/wiki/%D0%AE%D0%BD%D0%B8%D1%82-%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) модулей;
* [функциональное тестирование](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_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

Разработчик не может быть уверен в правильности написанного им кода до тех пор, пока не сработают абсолютно все тесты модулей разрабатываемой им системы. Тесты модулей (юнит-тесты) позволяют разработчикам убедиться в том, что каждый из них по отдельности работает корректно. Они также помогают другим разработчикам понять, зачем нужен тот или иной фрагмент кода, и как он функционирует — в ходе изучения кода тестов логика работы тестируемого кода становится понятной, так как видно, как он должен использоваться. Тесты модулей также позволяют разработчику без каких-либо опасений выполнять [рефакторинг](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3) (refactoring).

Функциональные тесты предназначены для тестирования функционирования логики, образуемой взаимодействием нескольких (часто — довольно внушительного размера) частей. Они менее детальны, чем юнит-тесты, но покрывают гораздо больше — то есть, у тестов, которые при своём выполнении затрагивают больший объём кода, шанс обнаружить какое-либо некорректное поведение, очевидно, больше. По этой причине в промышленном программировании написание функциональных тестов нередко имеет больший приоритет, чем написание юнит-тестов.

Подробнее рассмотрим основные приемы XP, которые мы будем использоать.

1. [Разработка через тестирование](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D1%87%D0%B5%D1%80%D0%B5%D0%B7_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) Для XP более приоритетным является подход, называемый TDD (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) *test-driven development* — [разработка через тестирование](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D1%87%D0%B5%D1%80%D0%B5%D0%B7_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). В соответствии с этим подходом сначала пишется тест, который изначально не проходит (так как логики, которую он должен проверять, ещё просто не существует), затем реализуется логика, необходимая для того, чтобы тест прошёл.

**- Добавление теста**

Неизбежно этот тест не будет проходить, поскольку соответствующий код ещё не написан. (Если же написанный тест прошёл, это означает, что либо предложенная «новая» функциональность уже существует, либо тест имеет недостатки.)

**- Запуск всех тестов: убедиться, что новые тесты не проходят**

**- Написать код**

Важно писать код, предназначенный именно для прохождения теста. Не следует добавлять лишней и, соответственно, не тестируемой функциональности.

**- Запуск всех тестов: убедиться, что все тесты проходят**

**- Рефакторинг**

[Рефакторинг](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3) — процесс изменения внутренней структуры программы, не затрагивающий её внешнего поведения и имеющий целью облегчить понимание её работы, устранить дублирование кода, облегчить внесение изменений в ближайшем будущем.

**- Повторить цикл**

1. **Непрерывная интеграция** (CI, [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) *Continuous Integration*) — это практика [разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%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), которая заключается в слиянии рабочих копий в общую основную ветвь разработки несколько раз в день и выполнении частых автоматизированных сборок проекта для скорейшего выявления и решения интеграционных проблем.

- Исходный код и всё, что необходимо для сборки и тестирования проекта, хранится в репозитории [системы управления версиями](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8);

- Операции копирования из репозитория, сборки и тестирования всего проекта автоматизированы и легко вызываются из внешней программы.

1. **Па́рное программи́рование** — техника [программирования](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), при которой [исходный код](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) создаётся парами людей, программирующих одну задачу, сидя за одним рабочим местом. Один [программист](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%81%D1%82) («ведущий») управляет компьютером и, в основном, думает над кодированием в деталях. Другой программист («штурман»[[1]](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%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#cite_note-1)) сосредоточен на картине в целом и непрерывно просматривает код, производимый первым программистом. Время от времени они меняются ролями, обычно, каждые полчаса.

**Пинг-понг программирование** — разновидность парного программирования, адаптированная для метода [разработки через тестирование](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D1%87%D0%B5%D1%80%D0%B5%D0%B7_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

Процесс написания кода становится похожим на игру в пинг-понг:

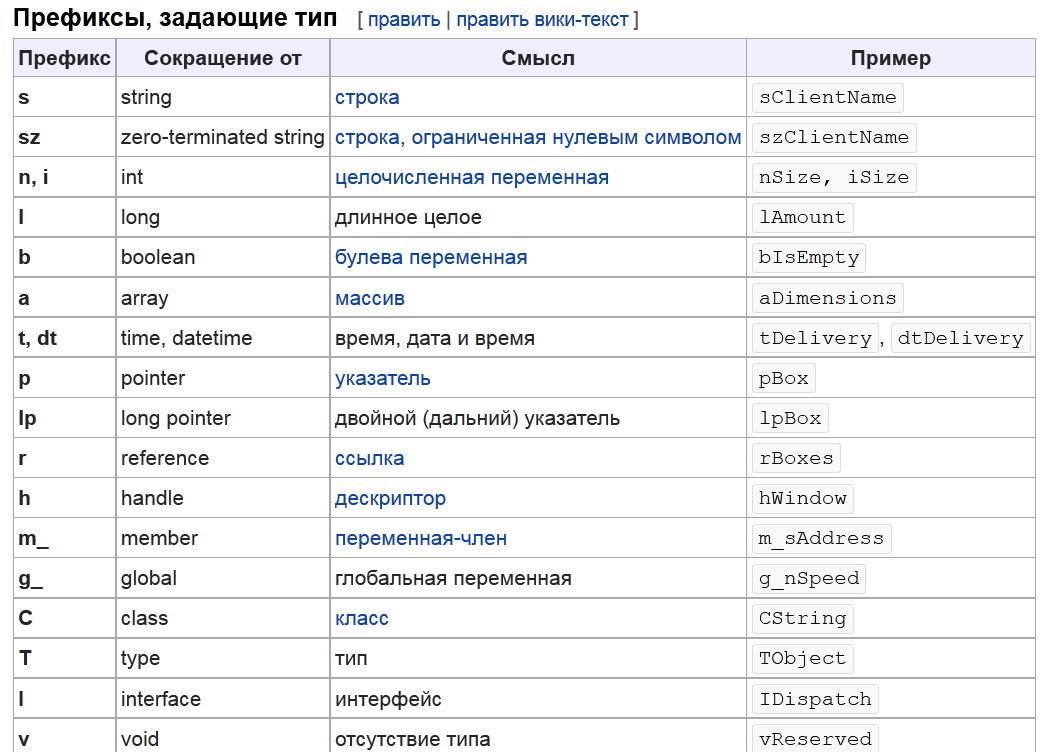
* первый участник пишет тест;
* второй — делает его проходящим, пишет ответный тест и отдаёт клавиатуру обратно.

1. **Станда́рт оформле́ния ко́да** (**станда́рт коди́рования**) ([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) *coding standards*, *coding convention*)

Обычно, стандарт оформления кода описывает:

* способы выбора названий и используемый регистр символов для имён переменных и других идентификаторов:
  + запись типа переменной в её идентификаторе ([венгерская нотация](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%BD%D0%B3%D0%B5%D1%80%D1%81%D0%BA%D0%B0%D1%8F_%D0%BD%D0%BE%D1%82%D0%B0%D1%86%D0%B8%D1%8F)) и

Рис. Венгерская нотация. Префиксы (пример)



* + регистр символов (нижний, верхний, «[верблюжий](https://ru.wikipedia.org/wiki/CamelCase)», «верблюжий» с малой буквы), использование [знаков подчёркивания](https://ru.wikipedia.org/wiki/%D0%97%D0%BD%D0%B0%D0%BA_%D0%BF%D0%BE%D0%B4%D1%87%D1%91%D1%80%D0%BA%D0%B8%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) для разделения слов;
* [стиль отступов](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B8%D0%BB%D1%8C_%D0%BE%D1%82%D1%81%D1%82%D1%83%D0%BF%D0%BE%D0%B2) при оформлении [логических блоков](https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%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%D0%B5) — используются ли [символы табуляции](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D0%B2%D0%BE%D0%BB_%D1%82%D0%B0%D0%B1%D1%83%D0%BB%D1%8F%D1%86%D0%B8%D0%B8), ширина отступа;
* способ расстановки скобок, ограничивающих логические блоки;
* использование пробелов при оформлении логических и арифметических выражений;
* стиль [комментариев](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%80%D0%B8%D0%B8_%28%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%29) и использование [документирующих комментариев](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D1%83%D1%8E%D1%89%D0%B8%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%80%D0%B8%D0%B9).

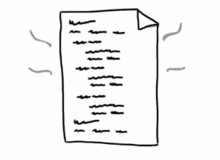
Вне стандарта подразумевается:

* отсутствие [магических чисел](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE_%28%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%29);
* ограничение размера кода по горизонтали (чтобы помещался на экране) и вертикали (чтобы весь код файла держался в памяти), а также функции или метода в размер одного экрана.

1. Рефакторинг

**Код с запашко́м** (*code smell*) — термин, обозначающий [код](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) с признаками (запахами) проблем в системе. Запахи кода[[⇨]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC" \l ".D0.97.D0.B0.D0.BF.D0.B0.D1.85.D0.B8_.D0.BA.D0.BE.D0.B4.D0.B0) — это ключевые признаки необходимости [рефакторинга](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3)[[4]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-Vigorous_Hive_CodeSmell.E2.80.94.E2.80.94.E2.80.94-4).

#### Дублирование кода

[](https://commons.wikimedia.org/wiki/File:Duplicate_code.gif?uselang=ru)

Пример трехкратного дублирования кода и избавления от этого запашка с помощью [выделения метода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3#.D0.92.D1.8B.D0.B4.D0.B5.D0.BB.D0.B5.D0.BD.D0.B8.D0.B5_.D0.BC.D0.B5.D1.82.D0.BE.D0.B4.D0.B0_.28Extract_Method.29) и последующих вызовов кода созданного метода

Основная статья: [**Дублирование кода**](https://ru.wikipedia.org/wiki/%D0%94%D1%83%D0%B1%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B0)

Дублирование кода — это использование одинаковых структур кода в нескольких местах. Объединение этих структур позволит улучшить программный код[[8]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9454-8).

Примеры дублирования и методы их устранения:

* Одно и то же выражение присутствует в двух методах одного и того же класса: необходимо применить [«Выделение метода» (Extract Method)](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3#.D0.92.D1.8B.D0.B4.D0.B5.D0.BB.D0.B5.D0.BD.D0.B8.D0.B5_.D0.BC.D0.B5.D1.82.D0.BE.D0.B4.D0.B0_.28Extract_Method.29) и вызывать код созданного метода из обеих точек;
* Одно и то же выражение есть в двух подклассах, находящихся на одном уровне: необходимо применить «Выделение метода» (Extract Method) для обоих классов с последующим «Подъемом поля» (Pull Up Field) или «Формированием шаблона метода» (Form Template Method), если код похож, но не совпадает полностью. Если оба метода делают одно и то же с помощью разных алгоритмов, можно выбрать более четкий из этих алгоритмов и применить «Замещение алгоритма» (Substitute Algorithm);
* Дублирующийся код находится в двух разных классах: необходимо применить «Выделение класса» (Extract Class) в одном классе, а затем использовать новый компонент в другом[[8]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9454-8).

#### Длинный метод

Среди объектных программ дольше всего живут программы с короткими [методами](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%28%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%29). Чем длиннее процедура, тем труднее её понять. Если у метода хорошее название, то не нужно смотреть его тело[[5]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.94.D1.83.D1.80.D0.BD.D0.BE_.D0.BF.D0.B0.D1.85.D0.BD.D1.83.D1.89.D0.B8.D0.B9_.D0.BA.D0.BE.D0.B4.E2.80.94.E2.80.94.E2.80.94-5).

Следует придерживаться эвристического правила: если ощущается необходимость что-то прокомментировать, нужно написать метод. Даже одну строку имеет смысл выделить в метод, если она нуждается в разъяснениях[[9]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9455-9).

* Для сокращения метода достаточно применить «Выделение метода» (Extract Method);
* Если локальные переменные и параметры препятствуют выделению метода, можно применить «Замену временной переменной вызовом метода» (Replace Temp with Query), «Введение граничного объекта» (Introduce Parametr Object) и «Сохранение всего объекта» (Preserve Whole Object)[[5]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.94.D1.83.D1.80.D0.BD.D0.BE_.D0.BF.D0.B0.D1.85.D0.BD.D1.83.D1.89.D0.B8.D0.B9_.D0.BA.D0.BE.D0.B4.E2.80.94.E2.80.94.E2.80.94-5);
* [Условные операторы](https://ru.wikipedia.org/wiki/%D0%A3%D1%81%D0%BB%D0%BE%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80) и [циклы](https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D0%BA%D0%BB_%28%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%29) свидетельствуют о возможности выделения в отдельный метод. Для работы с условными выражениями подходит «Декомпозиция условных операторов» (Decompose Conditional). Для работы с циклом — «Выделение метода» (Extract Method)[[9]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9455-9).

#### Большой класс

Когда [класс](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_%28%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%29) реализует слишком обширную функциональность, стоит подумать о вынесении некоторой части кода в подкласс. Это избавит разработчиков от чрезмерного количества имеющихся у класса атрибутов и дублирования кода[[9]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9455-9).

* Для уменьшения класса используется «Выделение класса» (Extract Class) или «Выделение подкласса» (Extract Subclass). При этом следует обращать внимание на общность в названии атрибутов и на то, использует ли класс их все одновременно[[5]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.94.D1.83.D1.80.D0.BD.D0.BE_.D0.BF.D0.B0.D1.85.D0.BD.D1.83.D1.89.D0.B8.D0.B9_.D0.BA.D0.BE.D0.B4.E2.80.94.E2.80.94.E2.80.94-5);
* Если большой класс является классом [GUI](https://ru.wikipedia.org/wiki/GUI), может потребоваться переместить его данные и поведение в отдельный объект предметной области. При этом может оказаться необходимым хранить копии некоторых данных в двух местах и обеспечить их согласованность. «Дублирование видимых данных» (Duplicate Observed Data) предлагает путь, которым можно это осуществить[[10]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9456-10).

#### Длинный список параметров

В длинных списках [параметров](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80_%28%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%29) трудно разбираться, они становятся противоречивыми и сложными в использовании. Использование [объектов](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_%28%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%29) позволяет, в случае изменения передаваемых данных, модифицировать только сам объект. Работая с объектами, следует передавать ровно столько, чтобы метод мог получить необходимые ему данные[[10]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9456-10).

* «Замена параметра вызовом метода» (Replace Parameter with Method) применяется, когда можно получить данные путём вызова метода объекта. Этот объект может быть полем или другим параметром.
* «Сохранение всего объекта» (Preserve Whole Object) позволяет взять группу данных, полученных от объекта, и заменить их самим объектом.
* «Введение граничного объекта» (Introduce Parameter Object) применяется, если есть несколько элементов данных без логического объекта[[10]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9456-10).

#### Расходящиеся модификации

Проблема возникает, когда при модификации в системе невозможно выделить определенное место, которое нужно изменить. Это является следствием плохой структурированности [ПО](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)[[10]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9456-10) или [программирования методом копирования-вставки](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_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BC_%D0%BA%D0%BE%D0%BF%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F-%D0%B2%D1%81%D1%82%D0%B0%D0%B2%D0%BA%D0%B8).

* Если набор методов необходимо изменять каждый раз при внесении определенных модификаций в код, то применяется «Выделение класса» (Extract Class) (Например, три метода меняются каждый раз когда подключается новая [БД](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), а четыре — при добавлении финансового инструмента)[[5]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.94.D1.83.D1.80.D0.BD.D0.BE_.D0.BF.D0.B0.D1.85.D0.BD.D1.83.D1.89.D0.B8.D0.B9_.D0.BA.D0.BE.D0.B4.E2.80.94.E2.80.94.E2.80.94-5).

#### Стрельба дробью

При выполнении любых модификаций приходится вносить множество мелких изменений в большое число классов. «Стрельба дробью» похожа на «Расходящуюся модификацию», но является её противоположностью. Расходящаяся модификация имеет место, когда есть один класс, в котором производится много различных изменений, а «Стрельба дробью» — это одно изменение, затрагивающее много классов[[11]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9456.E2.80.9457-11).

* Вынести все изменения в один класс позволят [«Перемещение метода» (Move Method)](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3#.D0.9F.D0.B5.D1.80.D0.B5.D0.BC.D0.B5.D1.89.D0.B5.D0.BD.D0.B8.D0.B5_.D0.BC.D0.B5.D1.82.D0.BE.D0.B4.D0.B0_.28Move_Method.29) и «Перемещение поля» (Move Field);
* Если нет подходящего класса, то следует создать новый класс;
* Если это необходимо, следует воспользоваться «Встраиванием класса» (Inline Class)[[5]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.94.D1.83.D1.80.D0.BD.D0.BE_.D0.BF.D0.B0.D1.85.D0.BD.D1.83.D1.89.D0.B8.D0.B9_.D0.BA.D0.BE.D0.B4.E2.80.94.E2.80.94.E2.80.94-5).

#### Жадные функции

Метод обращается к данным другого объекта чаще, чем к собственным данным[[5]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.94.D1.83.D1.80.D0.BD.D0.BE_.D0.BF.D0.B0.D1.85.D0.BD.D1.83.D1.89.D0.B8.D0.B9_.D0.BA.D0.BE.D0.B4.E2.80.94.E2.80.94.E2.80.94-5).

* «Перемещение метода» (Move Method) применяется, если метод явно следует перевести в другое место;
* «Выделение метода» (Extract Method) применяется к части метода, если только эта часть обращается к данным другого объекта;
* Метод использует функции нескольких классов: определяется, в каком классе находится больше всего данных, и метод помещается в класс вместе с этими данными, или с помощью «Выделения метода» (Extract Method) метод разбивается на несколько частей и они помещаются в разные места[[12]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9457-12).

Фундаментальное практическое правило гласит: то, что изменяется одновременно, надо хранить в одном месте. Данные и функции, использующие эти данные, обычно изменяются вместе, но бывают исключения[[12]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9457-12).

#### Группы данных

Группы данных, встречающихся совместно, нужно превращать в самостоятельный класс[[12]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9457-12).

* «Выделение метода» (Extract Method) используется для полей;
* «Введение граничного объекта» (Introduce Parameter Object) или «Сохранение всего объекта» (Preserve Whole Object) для параметров методов[[13]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.94.D1.83.D1.80.D0.BD.D0.BE_.D0.BF.D0.B0.D1.85.D0.BD.D1.83.D1.89.D0.B8.D0.B9_.D0.BA.D0.BE.D0.B4.E2.80.94.E2.80.94.E2.80.9457-13).

Хорошая проверка: удалить одно из значений данных и проверить, сохранят ли смысл остальные. Если нет, это верный признак того, что данные напрашиваются на объединение их в объект[[12]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9457-12).

#### Одержимость элементарными типами

Проблема связана с использованием элементарных типов вместо маленьких объектов для небольших задач, таких как валюта, диапазоны, специальные строки для телефонных номеров и т. п.

* «Замена значения данных объектом» (Replace Data Value with Object);
* «Замена массива объектом» (Replace Array with Object);
* Если это код типа, то используйте «Замену кода типа классом» (Replace Type Code with Class), «Замену кода типа подклассами» (Replace Type Code with Subclasses) или «Замену кода типа состоянием/стратегией» (Replace Type Code with State/Strategy)[[5]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.94.D1.83.D1.80.D0.BD.D0.BE_.D0.BF.D0.B0.D1.85.D0.BD.D1.83.D1.89.D0.B8.D0.B9_.D0.BA.D0.BE.D0.B4.E2.80.94.E2.80.94.E2.80.94-5).

#### Операторы типа switch

Одним из очевидных признаков объектно-ориентированного кода служит сравнительно редкое использование операторов типа [switch (или case)](https://ru.wikipedia.org/wiki/Switch_%28%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80%29#.D0.9F.D0.B5.D1.80.D0.B5.D0.BA.D0.BB.D1.8E.D1.87.D0.B0.D1.82.D0.B5.D0.BB.D1.8C). Часто один и тот же блок switch оказывается разбросанным по разным местам программы. При добавлении в переключатель нового варианта приходится искать все эти блоки switch и модифицировать их. Как правило, заметив блок switch, следует подумать о полиморфизме[[14]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9458-14).

* Если switch переключается по коду типа, то следует использовать «Замену кода типа подклассами» (Replace Type Code with Subclasses) или «Замену кода типа состоянием/стратегией» (Replace Type Code with State/Strategy);
* Может понадобиться «Выделение метода» (Extract Method) и «Перемещение метода» (Move Method) чтобы изолировать switch и поместить его в нужный класс;
* После настройки структуры наследования следует использовать [«Замену условного оператора полиморфизмом» (Replace Conditional with Polymorphism)](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3#.D0.97.D0.B0.D0.BC.D0.B5.D0.BD.D0.B0_.D1.83.D1.81.D0.BB.D0.BE.D0.B2.D0.BD.D0.BE.D0.B3.D0.BE_.D0.BE.D0.BF.D0.B5.D1.80.D0.B0.D1.82.D0.BE.D1.80.D0.B0_.D0.BF.D0.BE.D0.BB.D0.B8.D0.BC.D0.BE.D1.80.D1.84.D0.B8.D0.B7.D0.BC.D0.BE.D0.BC_.28Replace_Conditional_with_Polymorphism.29)[[5]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.94.D1.83.D1.80.D0.BD.D0.BE_.D0.BF.D0.B0.D1.85.D0.BD.D1.83.D1.89.D0.B8.D0.B9_.D0.BA.D0.BE.D0.B4.E2.80.94.E2.80.94.E2.80.94-5).

#### Параллельные иерархии наследования

В коде с таким запашком всякий раз при порождении подкласса одного из классов приходится создавать подкласс другого класса[[14]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9458-14).

* Общая стратегия устранения дублирования состоит в том, чтобы заставить экземпляры одной иерархии ссылаться на экземпляры другой иерархии, а затем убрать иерархию в ссылающемся классе c помощью «Перемещения метода» (Move Method) и «Перемещения поля» (Move Field)[[14]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9458-14).

#### Ленивый класс

Класс, затраты на существование которого не окупаются выполняемыми им функциями, должен быть ликвидирован[[14]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9458-14).

* При наличии подклассов с недостаточными функциями попробуйте «Свертывание иерархии» (Collapse Hierarchy);
* Почти бесполезные компоненты должны быть подвергнуты «Встраиванию класса» (Inline Class)[[14]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9458-14).

#### Теоретическая общность

Этот случай возникает когда на определенном этапе существования программы обеспечивается набор механизмов, который, возможно, потребуется для некоторой будущей функциональности. В итоге программу становится труднее понимать и сопровождать[[15]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9459-15).

* Для незадействованных абстрактных классов используйте «Сворачивание иерархии» (Collapse Hierarhy);
* Ненужная [делегация](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BB%D0%B5%D0%B3%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%28%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%29) может быть удалена с помощью «Встраивания класса» (Inline Class);
* Методы с неиспользуемыми параметрами должны быть подвергнуты «Удалению параметров» (Remove Parameter)[[5]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.94.D1.83.D1.80.D0.BD.D0.BE_.D0.BF.D0.B0.D1.85.D0.BD.D1.83.D1.89.D0.B8.D0.B9_.D0.BA.D0.BE.D0.B4.E2.80.94.E2.80.94.E2.80.94-5).

#### Временное поле

Временные поля — это поля, которые нужны объекту только при определенных обстоятельствах. Такое положение вещей трудно для понимания, так как ожидается, что объекту нужны все его поля[[16]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.92.D1.80.D0.B5.D0.BC.D0.B5.D0.BD.D0.BD.D0.BE.D0.B5_.D0.BF.D0.BE.D0.BB.D0.B5.E2.80.94.E2.80.94.E2.80.94-16).

* Временные поля и весь код, работающий с ними следует поместить в отдельный класс с помощью «Выделения класса» (Extract Class);
* Удалить условно выполняемый код можно с помощью «Введения объекта Null» (Introduce Null Object) для создания альтернативного компонента[[15]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9459-15).

#### Цепочка вызовов

Цепочка вызовов появляется тогда, когда клиент запрашивает у одного объекта другой объект, другой объект запрашивает еще один объект и т. д. Такие последовательности вызовов означают, что клиент связан с навигацией по структуре классов. Любые изменения промежуточных связей означают необходимость модификации клиента[[15]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9459-15).

* Для удаления цепочки вызовов применяется прием «Сокрытие делегирования» (Hide Delegate)[[15]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9459-15).

#### Посредник

Чрезмерное использование делегирования может привести к появлению классов, у которых большинство методов состоят только из вызова метода другого класса[[15]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9459-15).

* Если большую часть методов класс делегирует другому классу, нужно воспользоваться «Удалением посредника» (Remove Middle Man)[[17]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9460-17).

#### Неуместная близость

«Неуместная близость» возникает тогда, когда классы чаще, чем следовало бы, погружены в [закрытые](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_%28%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%29) части друг друга[[17]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9460-17).

* Избавиться от «Неуместной близости» можно с помощью «Перемещения метода» (Move Method) и «Перемещения поля» (Move Field);
* По возможности следует прибегнуть к «Замене двунаправленной связи однонаправленной» (Change Bidirectional Association to Unidirectional), «Выделению класса» (Extract Class) или воспользоваться «Сокрытием делегирования» (Hide Delegate)[[17]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9460-17).

#### Альтернативные классы с разными интерфейсами

Два класса, в которых часть функциональности общая, но методы, реализующие её, имеют разные параметры[[18]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3_.D0.BA.D0.BE.D0.B4.D0.B0.E2.80.94.E2.80.94.E2.80.94-18).

* Применяйте «Переименование метода» (Rename Method) ко всем методам, выполняющим одинаковые действия, но различающимся [сигнатурами](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%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%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9#.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)[[17]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9460-17).

#### Неполнота библиотечного класса

[Библиотеки](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_%28%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%29) через некоторое время перестают удовлетворять требованиям пользователей. Естественное решение — поменять кое-что в библиотеках, но библиотечные классы не изменять. Следует использовать методы рефакторинга, специально предназначенные для этой цели[[18]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3_.D0.BA.D0.BE.D0.B4.D0.B0.E2.80.94.E2.80.94.E2.80.94-18).

* Если надо добавить пару методов, используется «Введение внешнего метода» (Introduce Foreign Method);
* Если надо серьезно поменять поведение класса, используется «Введение локального расширения» (Introduce Local Extension)[[18]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3_.D0.BA.D0.BE.D0.B4.D0.B0.E2.80.94.E2.80.94.E2.80.94-18).

#### Классы данных

Классы данных это классы, которые содержат только поля и методы для доступа к ним, это просто контейнеры для данных, используемые другими классами[[18]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3_.D0.BA.D0.BE.D0.B4.D0.B0.E2.80.94.E2.80.94.E2.80.94-18).

* Следует применить «Инкапсуляцию поля» (Encapsulate Field) и «Инкапсуляцию коллекции» (Encapsulate Collection)[[5]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.94.D1.83.D1.80.D0.BD.D0.BE_.D0.BF.D0.B0.D1.85.D0.BD.D1.83.D1.89.D0.B8.D0.B9_.D0.BA.D0.BE.D0.B4.E2.80.94.E2.80.94.E2.80.94-5).

#### Отказ от наследства

Если наследник использует лишь малую часть унаследованных методов и свойств родителя это является признаком неправильной иерархии.

* Необходимо создать новый класс на одном уровне с потомком и с помощью «Спуска метода» (Push Down Method) и «Спуска поля» (Push Down Field) вытолкнуть в него все бездействующие методы. Благодаря этому в родительском классе будет содержаться только то, что используется совместно[[19]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9461-19).

#### Комментарии

Часто комментарии играют роль «дезодоранта» кода, который появляется в нем лишь потому, что код плохой. Почувствовав потребность написать комментарий, попробуйте изменить структуру кода так, чтобы любые комментарии стали излишними[[19]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9461-19).

* Если для объяснения действий блока все же требуется комментарий, попробуйте применить «Выделение метода» (Extract Method);
* Если метод уже выделен, но по-прежнему нужен комментарий для объяснения его действия, воспользуйтесь «Переименованием метода» (Rename Method);
* Если требуется изложить некоторые правила, касающиеся необходимого состояния системы, примените «Введение утверждения» (Introduce Assertion)[[19]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D1%81_%D0%B7%D0%B0%D0%BF%D0%B0%D1%88%D0%BA%D0%BE%D0%BC#cite_note-.D0.9C.D0.B0.D1.80.D1.82.D0.B8.D0.BD_.D0.A4.D0.B0.D1.83.D0.BB.D0.B5.D1.80_.D0.A0.D0.B5.D1.84.D0.B0.D0.BA.D1.82.D0.BE.D1.80.D0.B8.D0.BD.D0.B3.E2.80.942003.E2.80.94.E2.80.9461-19).