**Многопоточность или многонитевость**

[Java Core](https://javarush.ru/quests/QUEST_JAVA_CORE)

[Уровень 6](https://javarush.ru/quests/lectures?quest=QUEST_JAVA_CORE&level=6), Лекция 1

— Привет, Амиго! У нас новая и очень трудная тема. Сочувствую. Часто она считается одной из самых сложных не только в Java, но и в программировании вообще. Это – ***многонитевость (multithreading)***.

Представь себе средненькую компьютерную игру, например, гонки на звездолетах. Ты летишь по просторам космоса, уворачиваясь от метеоритов и патрульных крейсеров, а с тобой еще два десятка таких же участников этих нелегальных гонок.

Допустим, ты решил написать такую же игру. Твоей программе придется отслеживать команды управления (ввод с клавиатуры), перемещать звездолеты, рассчитывать их траектории и последствия столкновения, а также отрисовывать все это на экране пользователя. Это очень сложная работа.

Вспомни, как мы решили «проблему большой сложности» в том примере про рост курьерской компании.

— Мы разделили ее на независимые отделы и жестко задали (стандартизировали) способы их взаимодействия.

**— Но что делать, когда независимым частям нужно выполнить какой-то объем работы, параллельно с другими частями?! Ответ на этот вопрос – нити(трэды) (или как их неправильно называют – потоки).**

**Попробуй представить программу, как такого маленького робота, который бегает по коду и выполняет команды. Сначала выполнил команду, написанную в одной строке, затем перешел на следующую, и так далее.**

— Представил. Ничего сложного!

— Отлично. А теперь представь, что таких роботов у тебя несколько. И пока один занимается вводом от пользователя, второй меняет объекты в соответствии с ним. Третий выполняет код, по отображению этих объектов на экран, а четвертый несколько раз в секунду проверяет – не столкнулись ли корабли и, в случае столкновения, просчитывает его результаты.

Таким образом, мы можем разделить программу не только на независимые части/объекты, но и добиться того, что эти части будут выполнять свою работу независимо друг от друга. А чем меньше взаимодействия между отдельными частями, тем меньше сложность программы.

Представь, что **ты смог заменить менеджера – скриптом**, **рассылающим письма**. А остальные отделы компании об этом даже не догадались. Такие примеры уже имели место в 26 веке и показали отличные результаты. **Большинство менеджеров, и даже топ-менеджеров, может быть успешно заменено скриптом средней сложности.** Только после вмешательства «профсоюза офисного планктона» удалось остановить массовые увольнения менеджеров. Но это так – отвлечение от темы.

— Как интересно.

— Мало того, что в программе может быть несколько таких «маленьких роботов», исполняющих код, так эти роботы могут еще общаться друг с другом и порождать новых роботов.

— Порождать новых роботов?

— Да, для выполнения новых задач. Иногда выгодно создать еще одного робота (еще одну нить), которая будет выполнять какое-то действие одновременно с текущей нитью (роботом).

— Думаю, что это хорошая вещь, хоть пока не могу придумать, где бы я этим пользовался.

А почему это называется «нити»?

— Представь, что все роботы разного цвета, и каждый робот помечает своим цветом все команды, которые он выполнил. Таким образом, за маленьким роботом, как за карандашом, будет тянуться след. Этот след тянется за роботом, как нитка за иголкой.

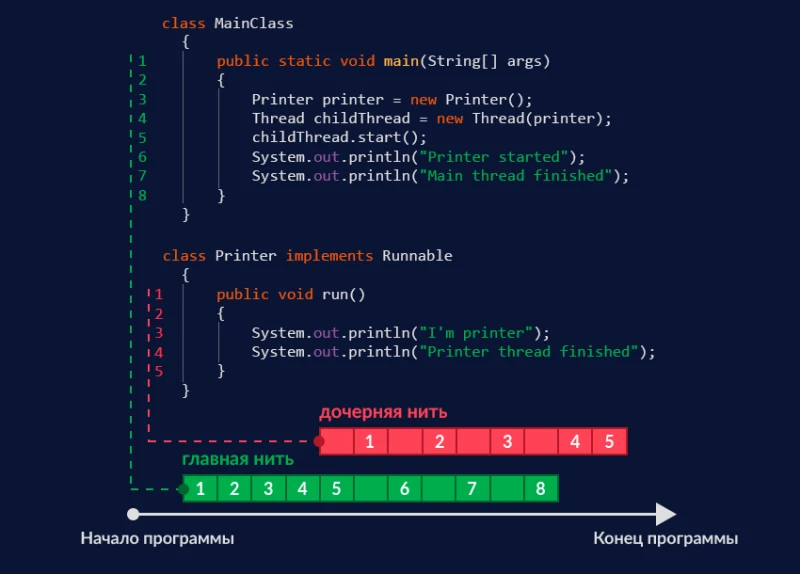
У каждого такого «маленького робота» есть задание, для исполнения которого его создали. И нить – это набор команд, выполненных в процессе исполнения этого задания.

Допустим, ты летишь на звездолете, чтобы доставить груз. Тогда «доставить груз» — это твое задание, ты в процессе его исполнения. А путь, который ты пролетел – это твоя нить. Можно сказать, что каждому новому заданию, каждой еще не решенной задаче соответствует своя нить – путь, который еще предстоит пройти.

— Другими словами, есть задание и «маленький робот», который его исполняет, а нить – это всего лишь взгляд на текущее положение дел со стороны?

— Именно так.

А вот как все это работает глубоко внутри. Т.к. процессор у компьютера один, то он может выполнять только одну команду одновременно. Поэтому происходит вот что: процессор постоянно переключается между нитями. Переключился на новую нить, выполнил несколько ее команд, затем переключился на следующую нить, выполнил несколько ее команд и так далее. Но т.к. переключение между нитями происходит сотни раз в секунду, со стороны кажется, что все нити работают одновременно.

[](https://cdn.javarush.ru/images/article/821343b2-82b0-4b58-9aef-840d7565d6b1/original.jpeg)