Блокирующие и неблокирующие

Сокеты и несокеты

Что такое блокирующая операция?

Речь не только про сокеты

Что такое блокирующая операция

Есть много где

- Операция, выполнение которой блокирует исполнение программы или ее части
- ВАЖНО: термин больше касается остановки одного или нескольких потоков, а не всего процесса
- ВАЖНО2: термин касается блокировок не только на уровне ядра операционной системы, но и на уровне виртуальных машин или в пользовательском пространстве
- Примеры?

Что такое блокирующая операция

Есть много где

- Операция, выполнение которой блокирует исполнение программы или ее части
- Примеры?
  - Мутексы, семафоры и условия
  - Операции с файловой системой
  - Операции с сокетами
  - Блокирующие коллекции

Что такое блокирующая операция

Есть много где

- Операция, выполнение которой блокирует исполнение программы или ее части
- Примеры?
  - Мутексы, семафоры и условия
  - Операции с файловой системой
  - Операции с сокетами
  - Блокирующие коллекции
  - Future, Barrier, Thread join, Sleep, Timer/Ticker

В чем преимущество?

Блокирующих операций

- Императивный (синхронный) код
  - Просто читать
  - Просто поддерживать
  - Просто понятно
  - Просто просто

```
public class Main {  Viacheslav Balashov +1 *
    public static void main(String[] args) throws Exception {
        int targetPort = 8880;
        var host = "192.168.50.15";

        try (var socket = new Socket(host, targetPort);
             var in = socket.getInputStream();
             var out = socket.getOutputStream();
        ) {
            var msg = "1".repeat( count: 10_000);
            var msgBytes = msg.getBytes();

            System.out.println(msgBytes.length);

            out.write(msgBytes);
            in.read();
            out.write(msgBytes);
            in.read();
            // etc
        }
    }
}
```

В чем недостатки?

Блокирующего подхода

- Блокируется поток — потенциальная потеря производительности
- А что если надо держать тысячи активных соединений (сокетов) и с каждым взаимодействовать? Создавать тысячу потоков?
- А если приложению надо делать помимо ожидания на сокете кучу полезной работы?
  - Например, перемножать матрицы?

Неужели нельзя использовать блокирующие?

Сокеты? Можно, но с оговорками

- Можно, если:
  - Если во время ожидания ничего не происходит
  - Простое приложение, где высоких нагрузок не будет, как и большого количества сокетов
- Нельзя, если:
  - Высоконагруженное приложение (все актуальные фреймворки и ЯП для бэкенда основаны на неблокирующих советах)

Что такое неблокирующая операция?

Тоже не только про сокеты

- Операция, исполнение которой не блокирует (приостанавливает) исполнение потока
- Если операция не может быть завершена, то она возвращает ошибку
- Примеры?

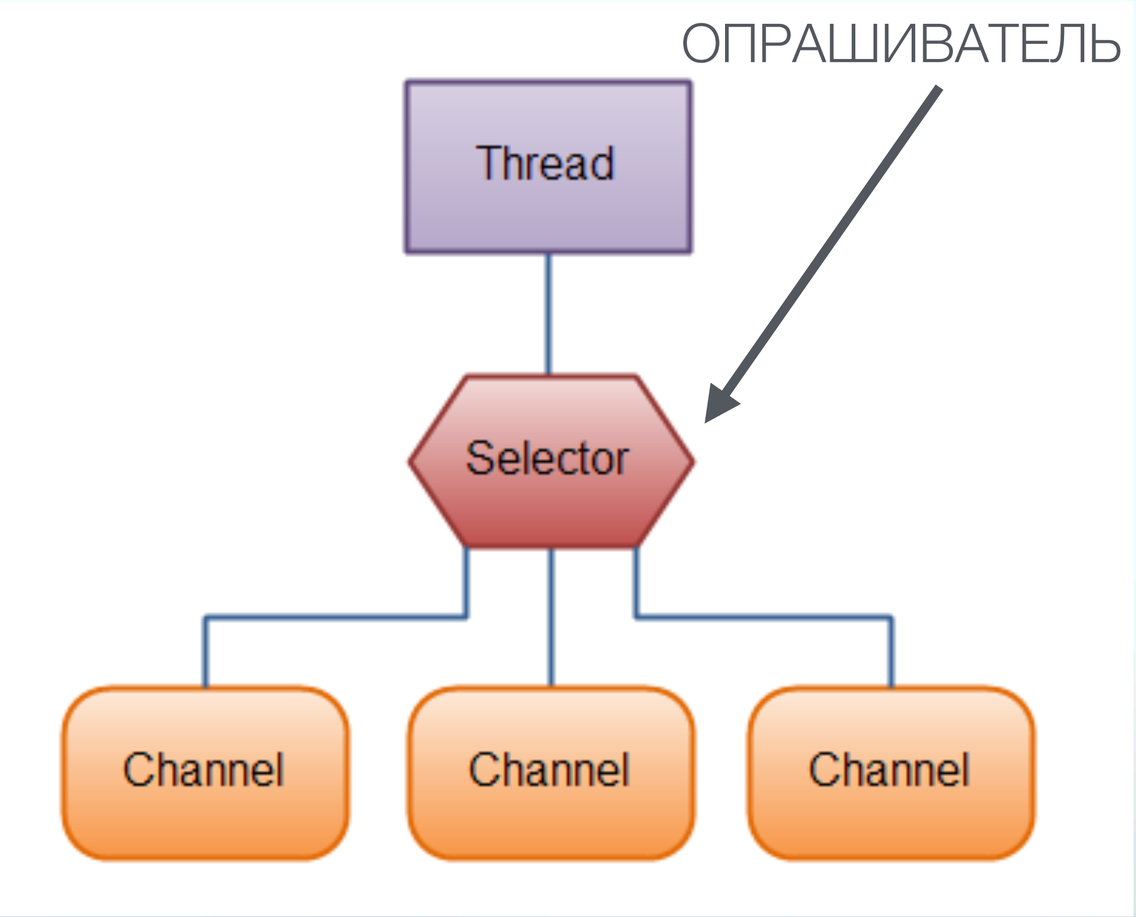
Примеры неблокирующих операций

Какие же?

- Атомарные локи и в целом взаимодействия с атомарными переменными
- Спин локи
- Неблокирующие сокеты
- Неблокирующие файловые операции
- Неблокирующие конкурентные коллекции

Как это происходит?

Во многих языках и фреймворках: под капотом, но не в Java SDK




```java
public static void anotherMain() throws Exception {  no usages
    var selector = Selector.open();

    var channel = ServerSocketChannel.open();
    channel.configureBlocking(false);
    channel.bind(new InetSocketAddress( port: 5000));

    channel.register(selector, SelectionKey.OP_ACCEPT);

    while (true) {
        int result = selector.select();
        if (result == 0) {
            continue;
        }

        selector.selectedKeys().forEach( SelectionKey key -> {
            // handle channel
        });
    }
}
```

В чем преимущество?

Неблокирующих операций

- Гораздо быстрее блокирующих
- Блокируется либо 0, либо 1 поток
- Оптимизации на уровне операционной системы
- Гораздо меньше потоков операционной системы используется
- Можно спокойно заниматься вычислительной работой
- Можно делить один thread pool на IO и non-IO операции
- Реализовано почти во всех современных платформах

В чем недостатки?

Блокирующего подхода

- Сложность кода возрастает в разы (и то не везде)
  - Асинхронность, события и т. д.
  - Теперь есть события (ACCEPT, CONNECT, READ)
- Сложнее отладка