Теория к лабе 6.

InetAddress

InetSocketAddress

Клиент

● Может работать на любом хосте (сервер не знает, где именно)

● Порт выбирается при отправлении запроса (первый свободный)

● Посылает запрос серверу, ждет ответ

Запрос

● Содержит информацию о клиенте и данные

Сервер

● Работает на определенном хосте (известный IP-адрес)

● Прослушивает определенный порт (зависит от сервиса)

● Ждет запрос от клиента, посылает ответ

Ответ

● Содержит данные

DatagramPacket

DatagramSocket

DatargramChannel

Класс java.util.ServiceLoader<Service>

* java.nio.file.spi.FileSystemProvider
* java.nio.channels.spi.AsynchronousChannelProvider
* java.nio.channels.spi.SelectorProvider
* java.nio.charset.spi.CharsetProvider
* java.text.spi.DateTimeFormatProvider
* java.text.spi.NumberFormatProvider
* java.util.spi.CalendarDataProvider
* java.sql.DriverManager

java.net.Socket — сокет для обмена (клиент и сервер)

● new Socket (адрес + порт — для отправления)

● Socket ServerSocket.accept() - для получения

● java.net.ServerSocket — фабрика сокетов

● new ServerSocket(порт) — на стороне сервера

● обмен данными через потоки ввода-вывода

● Socket.getInputStream()

● Socket.getOutputStream()

Протокол TCP

● ServerSocketChannel

◊ open()

◊ bind(SocketAddress local)

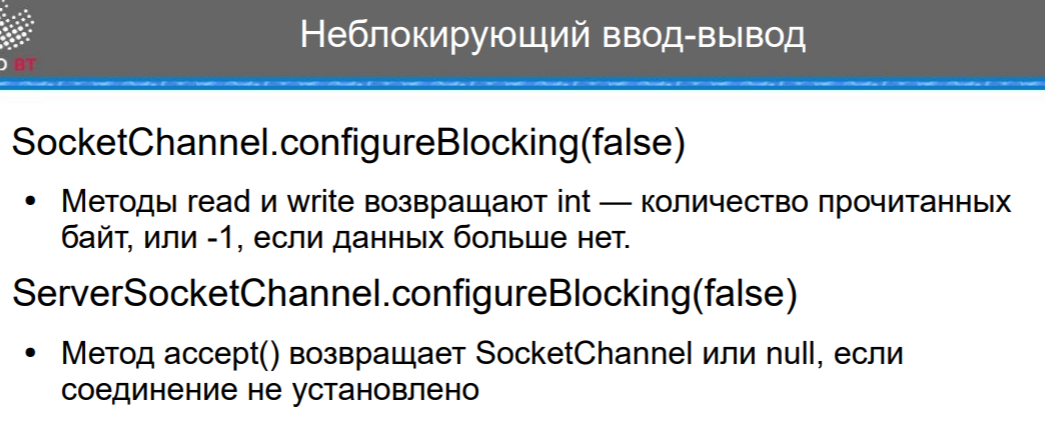
◊ SocketChannel accept()

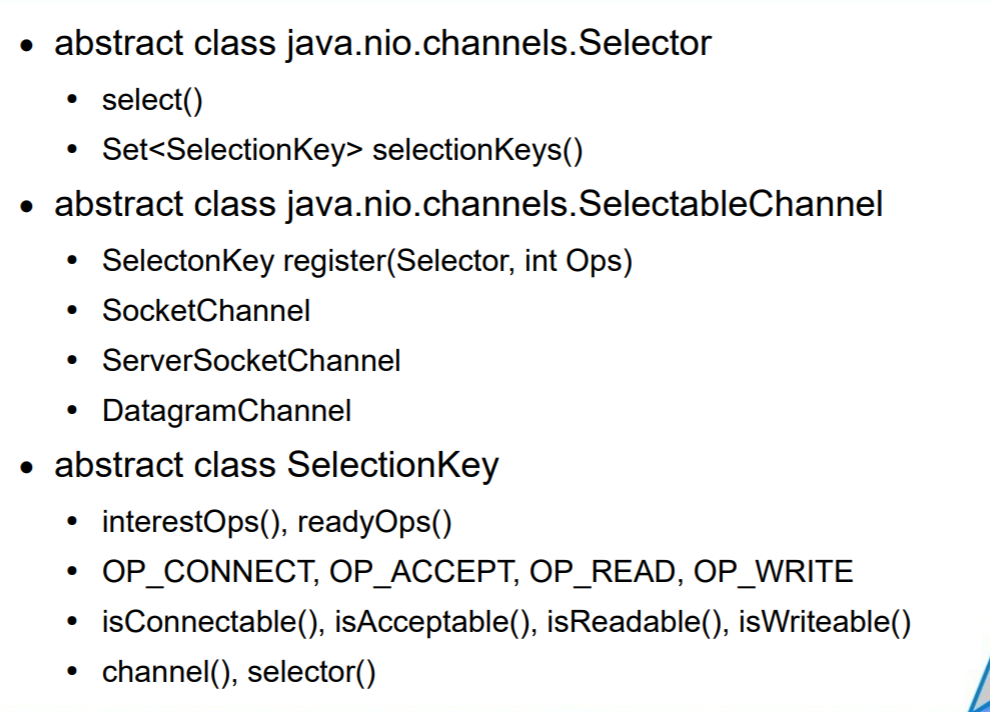
● SocketChannel

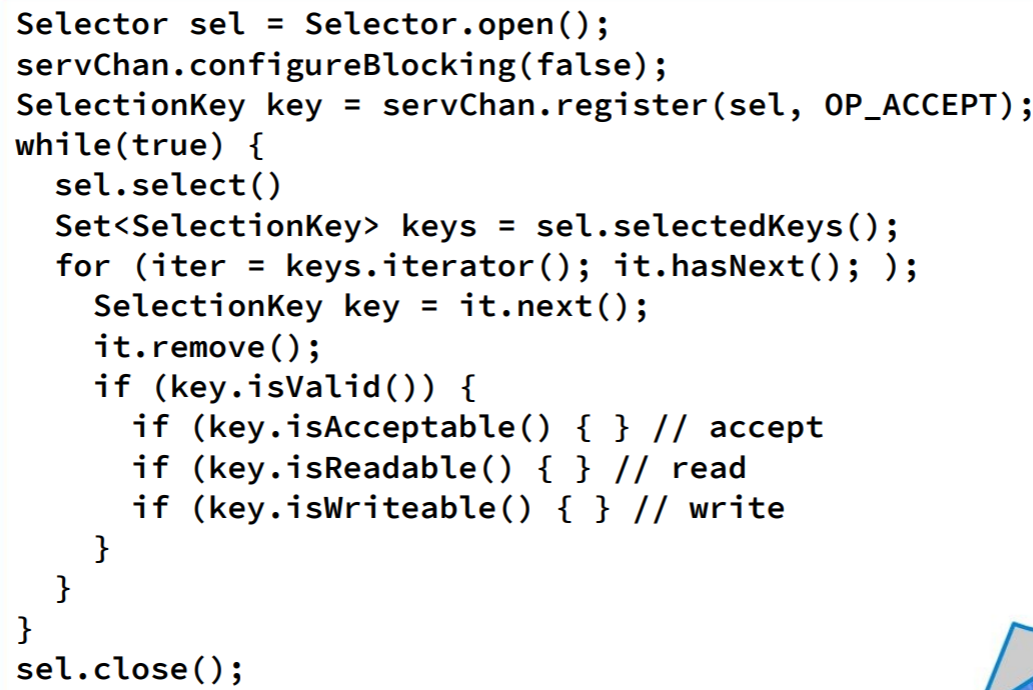
◊ open(SocketAddress remote)

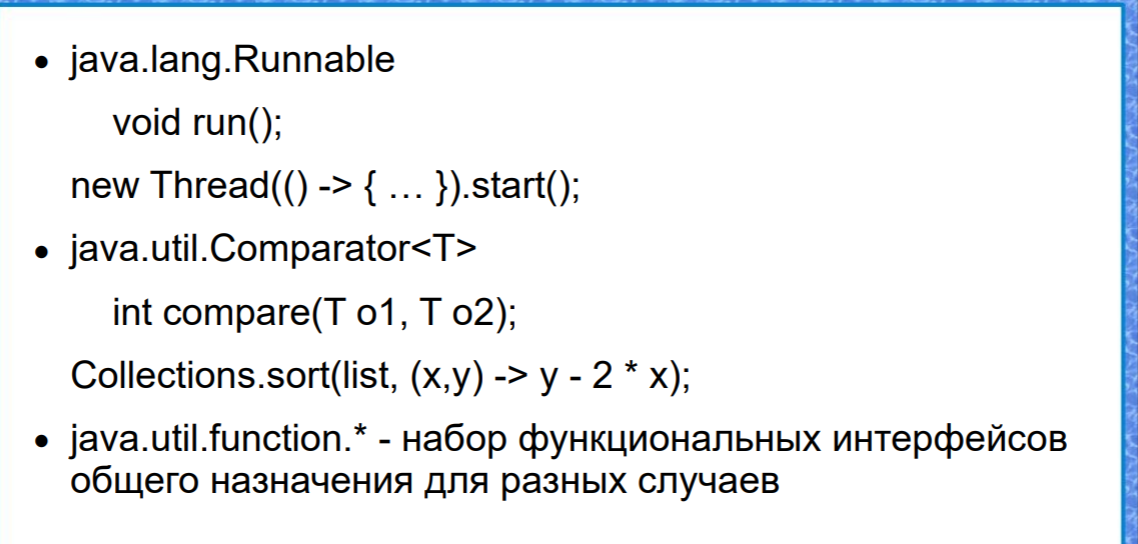
◊ write(ByteBuffer)

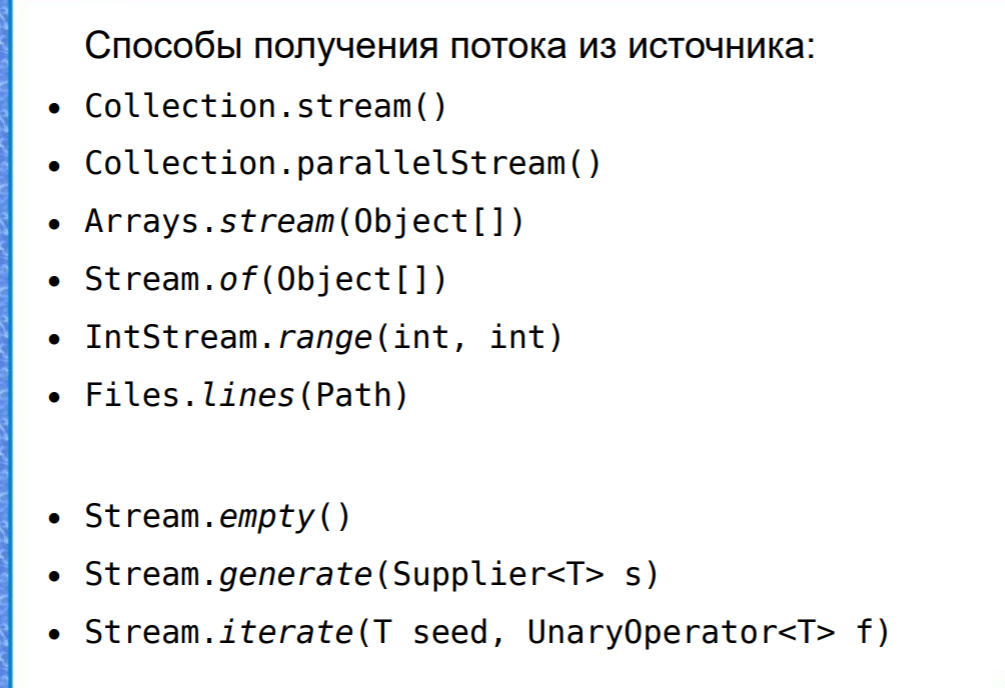
◊ read(ByteBuffer)

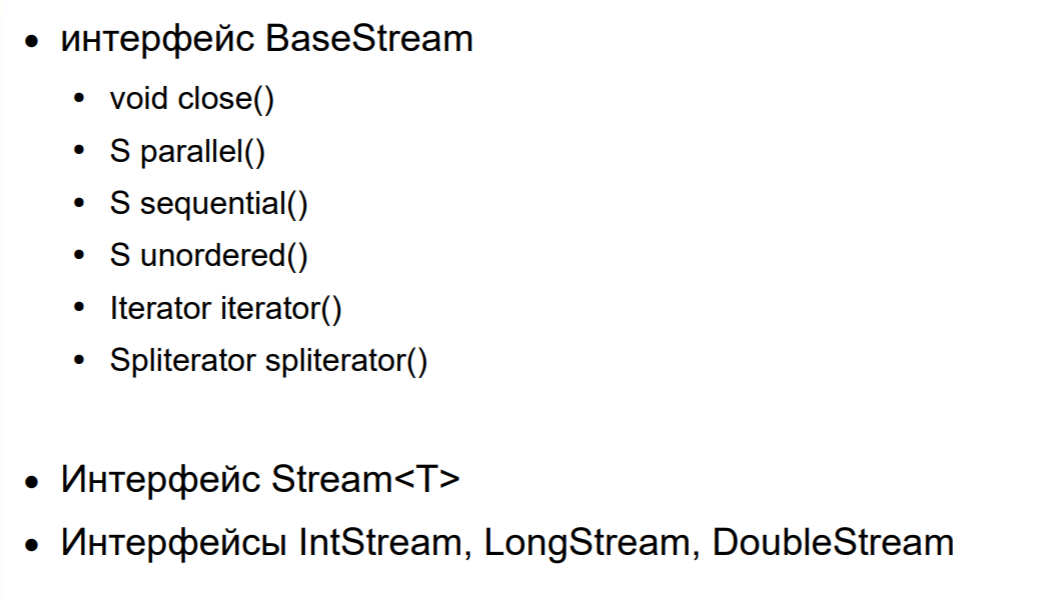


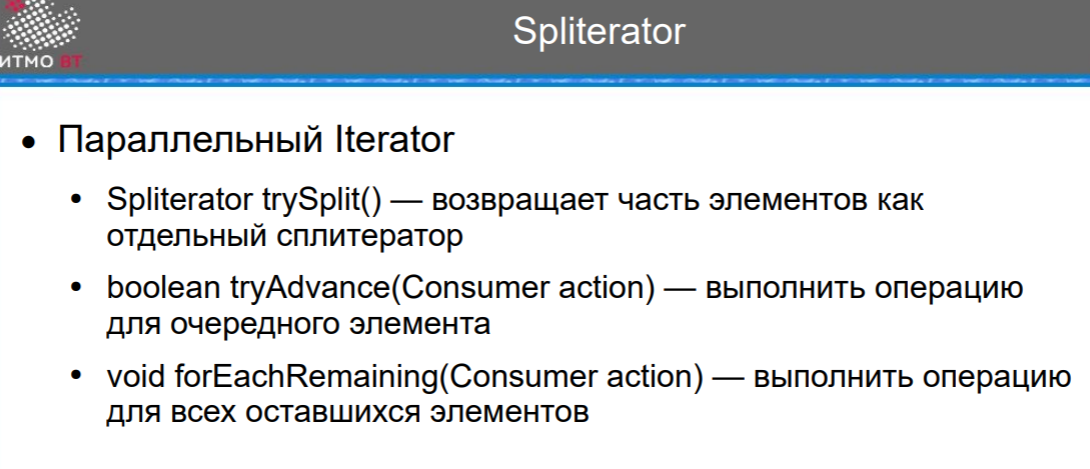






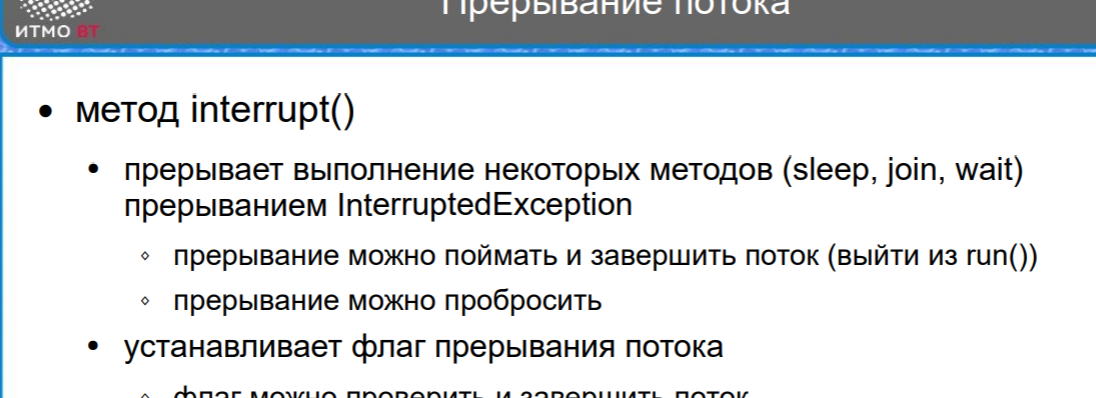


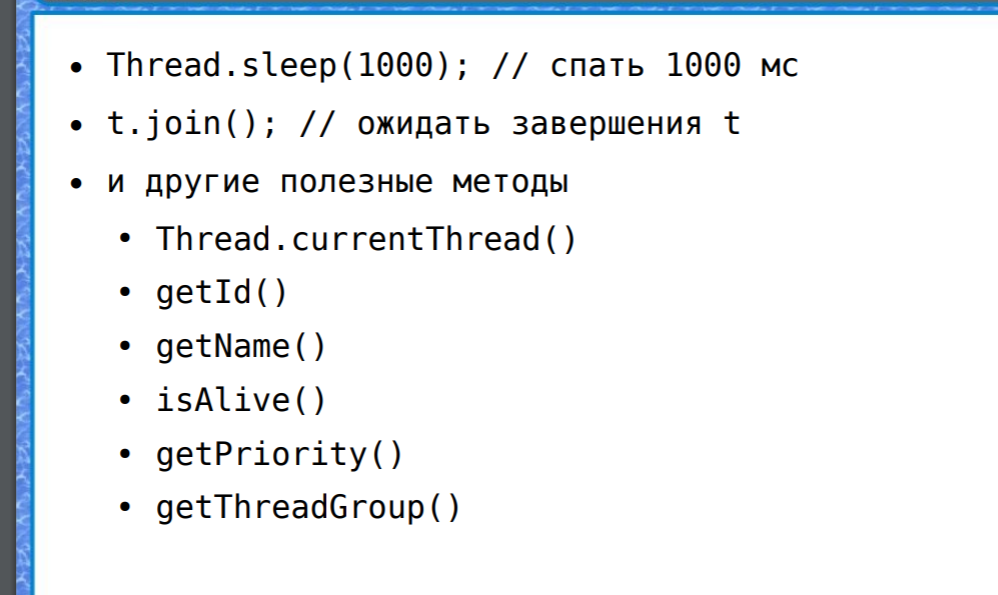


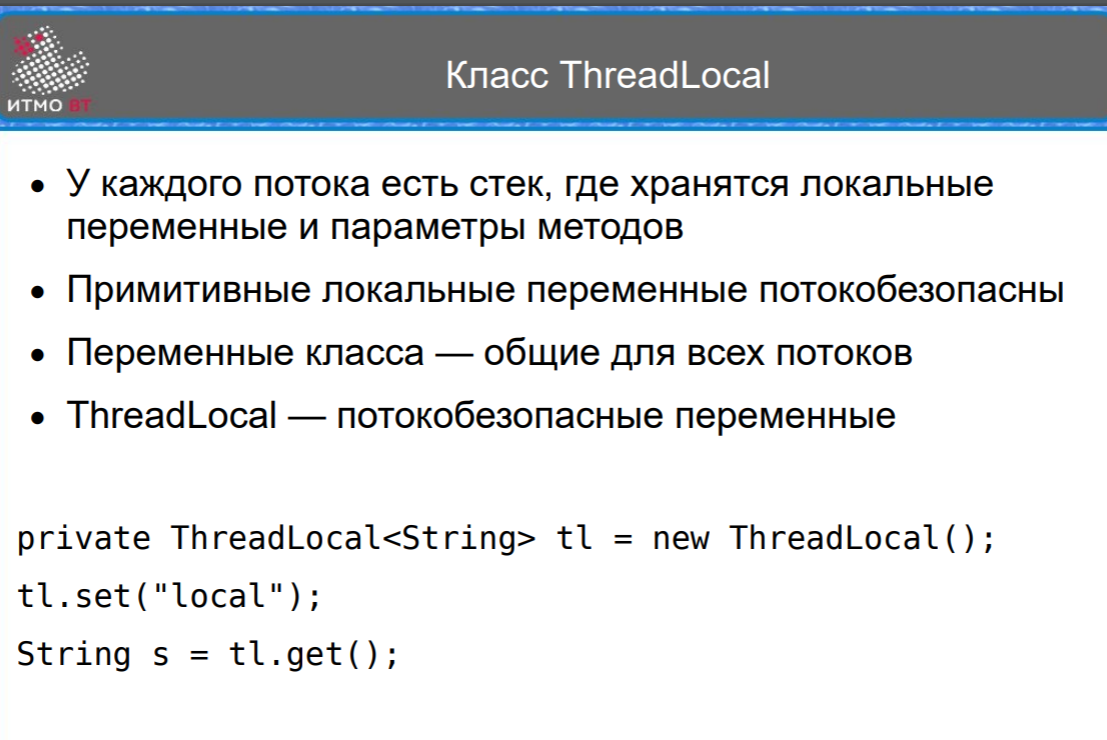


runnable

thread







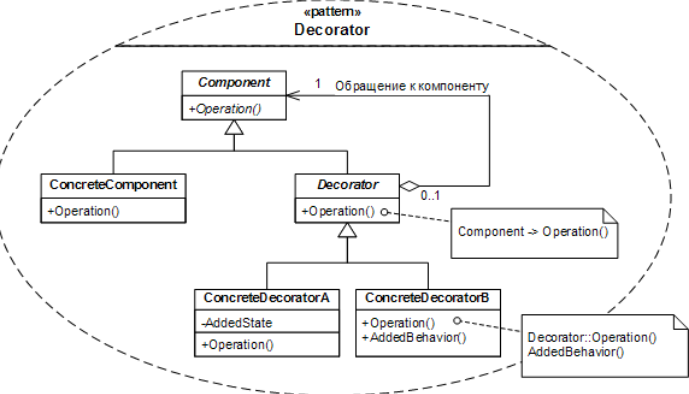


java.concurrent.\*

Decorator

Декоратор — это структурный паттерн, который позволяет добавлять объектам новые поведения на лету, помещая их в объекты-обёртки.

Декоратор позволяет оборачивать объекты бесчисленное количество раз благодаря тому, что и обёртки, и реальные оборачиваемые объекты имеют общий интерфейс.

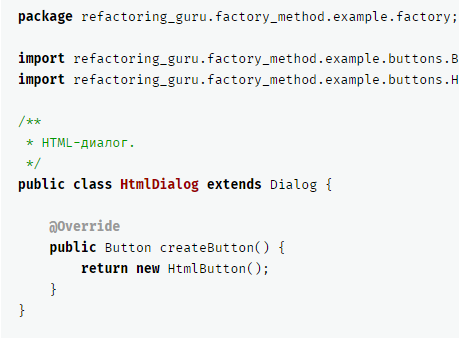


<https://refactoring.guru/ru/design-patterns/decorator/java/example>

Iterator

Просто пишем итератор. Можно указать методы hasNext, next, reset и т.п. В коде написать их реализацию.

Factory Method

Command

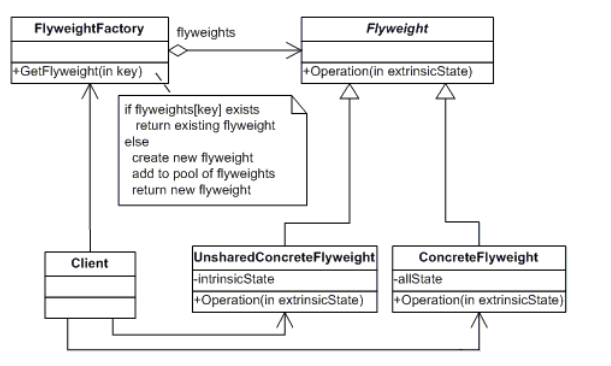
<https://javarush.ru/groups/posts/1684-pattern-command-svoimi-slovami>



Flyweight

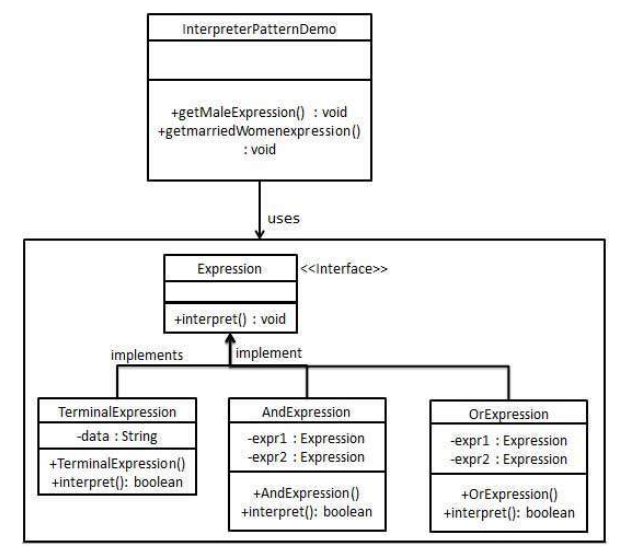
Легковес — это структурный паттерн, который экономит память, благодаря разделению общего состояния, вынесенного в один объект, между множеством объектов.

Легковес позволяет экономить память, кешируя одинаковые данные, используемые в разных объектах.



Interpreter

<https://www.tutorialspoint.com/design_pattern/interpreter_pattern.htm>



Singleton

<https://refactoring.guru/ru/design-patterns/singleton/java/example>

Strategy Java

<https://refactoring.guru/ru/design-patterns/strategy/java/example>

Пример: выбор оплаты картой или наличными через один интефейс, который выполняет функцию pay.

Adapter

Адаптер — это структурный паттерн, который позволяет подружить несовместимые объекты.

Адаптер выступает прослойкой между двумя объектами, превращая вызовы одного в вызовы понятные другому.

<https://refactoring.guru/ru/design-patterns/adapter/java/example>

Façade

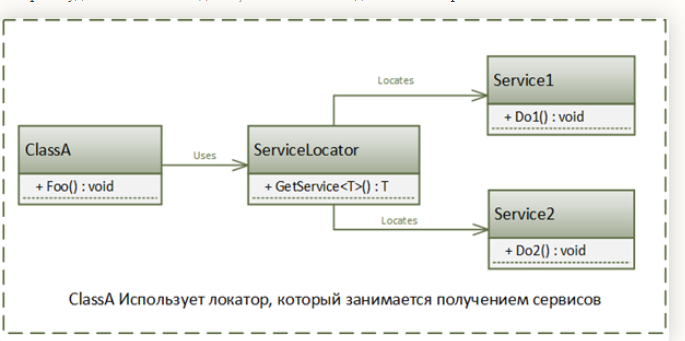
Фасад — это структурный паттерн, который предоставляет простой (но урезанный) интерфейс к сложной системе объектов, библиотеке или фреймворку.

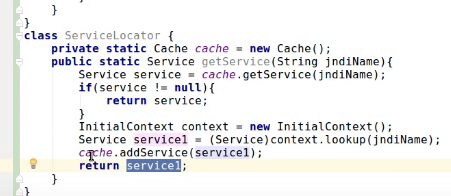
Кроме того, что Фасад позволяет снизить общую сложность программы, он также помогает вынести код, зависимый от внешней системы в единственное место.

Proxy

Заместитель — это объект, который выступает прослойкой между клиентом и реальным сервисным объектом. Заместитель получает вызовы от клиента, выполняет свою функцию (контроль доступа, кеширование, изменение запроса и прочее), а затем передаёт вызов сервисному объекту.

ServiceLocator





Как фабрика (сингтонская), можно организовать кеширование, чтобы не создавать кучу сервисов

Client-Server

Мо́дульное программи́рование — это организация программы как совокупности небольших независимых блоков, называемых модулями, структура и поведение которых подчиняются определённым правилам.[1] Использование модульного программирования позволяет упростить тестирование программы и обнаружение ошибок. Аппаратно-зависимые подзадачи могут быть строго отделены от других подзадач, что улучшает мобильность создаваемых программ.

Мо́дуль — функционально законченный фрагмент программы. Во многих языках (но далеко не обязательно) оформляется в виде отдельного файла с исходным кодом или поименованной непрерывной её части. Некоторые языки предусматривают объединение модулей в пакеты.

STREAM API

<https://habr.com/ru/company/luxoft/blog/270383/>

Сокет — это программная (логическая) конечная точка, устанавливающая двунаправленную коммуникацию между сервером и одной или несколькими клиентскими программами

UDP

<https://medium.com/nuances-of-programming/%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D1%8B-%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-udp-%D1%81%D0%BE%D0%BA%D0%B5%D1%82%D0%BE%D0%B2-%D0%BD%D0%B0-java-805ba96693df>

