**Задание.**

Написать DNS прокси-сервер с поддержкой "черного" списка доменных имен.

1. Для параметров используется конфигурационный файл, считывающийся при запуске сервера;

2. "Черный" список доменных имен находится в конфигурационном файле;

3. Адрес вышестоящего сервера также находится в конфигурационном файле;

4. Сервер принимает запросы DNS-клиентов на стандартном порту;

5. Если запрос содержит доменное имя, включенное в "черный" список, сервер возвращает клиенту ответ, заданный конфигурационным файлом (варианты: not resolved, адрес в локальной сети, ...).

6. Если запрос содержит доменное имя, не входящее в "черный" список, сервер перенаправляет запрос вышестоящему серверу, дожидается ответа и возвращает его клиенту.

Язык разработки: java

Использование готовых библиотек: без ограничений.

Использованный чужой код должен быть помечен соответствующими копирайтами, нарушать авторские права запрещено.

Остальные условия/допущения, не затронутые в тестовом задании - на усмотрение кандидата.

**Моя реализация.**

Ссылка на github

<https://github.com/hramovalexey/DNS-server>

Сервер реализован, как консольное приложение без интерфейса. Дополнительно к настройкам, обозначенным в задании, в конфигурационном файле указываются:

- время тайм-аута ожидания ответа от вышестоящего сервера,

- текст сообщения, отправляемого клиенту в случае тайм-аута.

Настройки хранятся в файле dnscfg.properties.

Сервер запускается, как экземпляр класса DnsServer.

Сервер работает в многопоточном режиме.

Для демонстрации работоспособности написан класс клиента TestClient.java, отправляющий тестовый DNS-запрос запущенному серверу.

Все пункты задания выполнены.

Кроме того, приложение парсит полностью заголовок DNS сообщения, хотя в текущей реализации не все полученные данные используются. Но в дальнейшем их можно было бы использовать для кэширования ответов и ведения лога (в задании таких требований не было, а на разработку надо время, поэтому я не стал это реализовывать).

Также не успел реализовать эффективный поиск среди запрещенных адресов. Пришлось реализовать обычный обход всего списка, хотя, считаю, что при большом списке и большом количестве запросов к серверу это может сказаться на производительности. Здесь можно было бы применить перфиксное дерево (Tries, n-Tries) для хранения списка адресов, но, как я уже сказал немножко не укладываюсь по времени.

Использованы только стандартные библиотеки из java.net, java.io, java.util и др.

Среда разработки IntelliJ IDEA.

Исходный код в папке src.