**Име:** Ралица Вунцова **Фн:** 5MI3400304

**Имейл**: [ralitsavun@gmail.com](mailto:ralitsavun@gmail.com) **Начална година**: 2023

**Програма:** магистър, (ЗИКСМ) **Курс:** 1

**Предмет**: Мрежово програмиране с Java **Дата:** 2023-01-18

**Преподавател:** проф. д-р Милен Петров

**Тема**: Транслитериращи услуги

## **1. Условие**

Проектът трябва да включва TCP сървър, който да извършва различни транслитериращи услуги, подобно на Google Translate. Освен това, трябва да бъде разработен и TCP client, който да си комуникира с TCP сървъра.

## **2. Въведение**

В днешно време хората все повече им се налага да използват различни езици. Съществуват много онлайн преводачи. Всеки знае и е използвал Google Translate. Но всъщност не много транслитериращи услуги, които да извършват по-специфични преобразувания, или предлагат работа с файлове и т.н., могат да се намерят в Интернет. Затова смятам, че идеята за разработване на система, която да позволява на потребителите да се възползват от разнообразни транслитериращи услуги, е много добра.

**3. Теория**

Проектът включва TCP клиент, който изпраща дума на сървъра, която трябва да бъде преведена, включително предоставя езика, от който иска да бъде преведена и езика, на който иска да бъде преведена, и TCP сървър, който ще ползва Apertium API (open-source translation platform), за да преведе думата на определения език и след това ще върне превода на клиента. Освен това, друг тип клиент може да изпрати файл със списък от думи, в който е възможно някои думи вече да са преведени. Сървърът трябва да върне файла обратно с всички думи преведени. Дефиниран по този начин проектът включва поне три теми от курса - потоци, работа с URL/URI и TCP сокети.

## **4. Използвани технологии**

Проектът представлява конзолно приложение, което се състои от различни клиенти и един сървър, като за комуникацията между тях се използват TCP сокети. Програмният език, който е използван е Java, версия 14. Използваното IDE е IntelliJ Idea.

## **5. Инсталация и настройки**

Предварително се създават следните файлове в директорията E:/project:

* to\_be\_translated\_en\_es.txt
* to\_be\_translated\_es\_en\_with\_translated\_words.txt
* to\_be\_translated\_mkd\_bul.txt

Съдържанието им може да се намери в в зип-а, предаден в заданието, или по-долу в документацията.

Преди да се стартира проектът, трябва да бъдат добавени следните jar файлове:

* commons-codec-1.10.jar
* commons-logging-1.2.jar
* httpclient-4.5.6.jar
* httpcore-4.4.10.jar
* json-simple-1.1.jar

Това може да стане като чрез File -> Project Structure -> Modules -> + -> JARs or directories. Тези файлове могат да бъдат открити в зип-а, предаден в заданието. Те са необходими за обработването на http заявките, които сървъра прави към Apertium API-то, за да получи превод за определена дума.

Последно, целият проект се билдва.

## **6. Кратко ръководство на потребителя**

За да се стартира сървъра - се стартира main метода, който се намира в клас Main в директория server. Следващата стъпка е да се стартира main метод, който се намира в клас Main в директория client. След това се появяват указания за последващи стъпки:

* Ако потребителят иска да се преведе само една дума – трябва да натисне 1.
  + Потребителят въвежда думата, която иска да бъде преведена.
  + Потребителят въвежда езика, от който иска да бъде преведена думата.
  + Потребителят въвежда езика, на който иска да бъде преведена думата.
  + След това потребителят получава превод на думата.
* Ако потребителят иска да се преведат думите в един файл – трябва да натисне 2.
  + Потребителят въвежда пътя на файла, който съдържа думи, които трябва да бъдат преведени.
    - Файлът трябва да съдържа на всеки ред дума, или дума и превод в следния формат:
      * *word\_to\_be\_translated=*
      * *already\_translated\_word=translation*
  + Потребителят въвежда езика, от който иска да бъдат преведени думите.
  + Потребителят въвежда езика, на който иска да бъдат преведени думите.
  + След това потребителят получава преведения файл.
    - Той ще бъде създаден в същата директория и със същото име + постфикс *“\_new”*
* Ако потребителят иска да спре приложението – натискаме 3.
  + Приложението спира да работи.

## **7. Примерни данни**

Следните данни могат да се подават в конзолата след като се стартират двата мейн метода:

Пример 1:

1

Cat

en

es

Пример 2:

1

Dog

en

es

Пример 3:

1

Mother

en

es

Пример 4:

1

Perro

es

en

Пример 5:

1

нога

mkd

bul

Пример 6:

2

E:/project/to\_be\_translated\_en\_es.txt

en

es

Пример 7:

2

E:/project/to\_be\_translated\_es\_en\_with\_translated\_words.txt

es

en

Пример 8:

2

E:/project/to\_be\_translated\_mkd\_bul.txt

mkd

bul

Пример 9:

Invalid input

Пример 10:

3

Съдържание на to\_be\_translated\_en\_es.txt:

Cat=

Dog=

Fox=

Tree=

Leaf=

Mother=

Father=

Sister=

Skin=

Fruit=

Съдържание на to\_be\_translated\_es\_en\_with\_translated\_words.txt:

Gato=

Perro=Dog

Fox=

Árbol=

Hoja=Leaf

Madre=

Padre=

Hermana=Sister

Piel=

Fruta=

Съдържание на to\_be\_translated\_mkd\_bul.txt:

заменик=

вредност=

ветеран=

редица=

мајка=

татко=

киднапирање=

рака=

нога=

надеж=

## **8. Описание на програмния код**

Структурата на проекта е следната:

* client
  + WordClient
  + FileClient
  + Main
* server
  + TranslatorServer
  + ConnectionHandler
  + WordHandler
  + FileHandler
  + ApertiumTranslator
  + Main

В директорията client се съдържа клиентската логика. Класът Main съдържа единствен метод main(), който има за цел да пита потребителя каква услуга иска да използва. В зависимост от изборът му – се създават WordClient или FileClient. WordClient се създава, когато потребителят иска да преведе само една дума. Той създава сокет и започва да пише в него – записва записва стрингът “word”, думата, езикът, от който трябва да се преведе и езикът, на който трябва да се преведе. След това чете от сокета, като това, което прочита е резултатът, върнат от сървъра, т.е. преведената дума. FileClient се създава, когато потребителят иска да преведе файл с много думи. Той създава сокет и започва да пише в него – записва стрингът “file” размера на файла и след това ред по ред записва съдържанието на файла. След това чете от сокета, като това, което прочита е резултатът, върнат от сървъра, т.е. файлът с думите, но преведен, като отново прочита размера на файла и после чете съдържанието на файла ред по ред. Новият файл се записва в същата директория, но с постфикс *“\_new”*.

В директорията server се съдържа сървърната логика. Класът Main съдържа единствен метод main(), който има за цел стартира сървъра. TranslationServer създава сокет и е имплементиран по такъв начин, че може да приема множество заявки от множество клиенти – за всяка отделна заявка се стартира нова нишка. Ролята на ConnectionHandler-a е да обработва заявките. Първоначално прочита един стринг – “word” или “file” и по този начин определя дали клиентът е WordClient или FileClient. В първия случай се създава WordHandler, във втория случай се създава FileHandler. WordHandler-а прочита езика, от който трябва да се преведе думата и езика, на който трябва да се преведе думата. След това извиква методът translateWord на ApertiumTranslator, който превежда думата. Последно, преведения стринг се записва в сокета. FileHandler-a прочита езика, от който трябва да се преведе думата и езика, на който трябва да се преведе думата, създава temporary файл, в който запазва данните от файла, изпратен от клиента и вкарва двойките думи в мап за удобство. След това за всяка думичка, която си няма превод извиква translateWord на ApertiumTranslator. Накрая създава наново структурата на файла, вече с преведените думи и го записва в сокета. Това, което всъщност translateWord на ApertiumTranslator прави е да изпрати пост заявка до API-то, подавайки думата и двата езика, да получи отговор, да го преобразува и да ретърне стринга, т.е. преведената дума.

## **9. Приноси на студента, ограничения и възможности за бъдещо разширение**

В момента в проекта съществуват ограничението, че Apertium API-то може да се използва само за определени двойки езици и дори и за тях много от преводите не са добри. Това може да се подобри като се смени и се използва друго по-добро API, като например Google Translate API. Също така, могат да се добавят функционалности за превеждане на цял текст, както и трансформации от следния вид: цифри/числа да се изписват като думи на съответния език.

## **10. Какво научих**

Успях да си усъвършенствам знанията си по сокети и нишки, както и упражних още веднъж работата с входно-изходни потоци.

Научих се да правя HTTP заявки в JAVA.

## **11. Използвани източници**

[1] Решения на проблеми, възникнали по време на имплементацията <https://stackoverflow.com/>

[2] HTTP POST заявки в Java <https://www.baeldung.com/httpclient-post-http-request>

[2] Apertium документацията <https://wiki.apertium.org/wiki/Main_Page>

[3] Apertium платформата <https://apertium.org>

Предал (подпис): ………………………….

/*фн, име*/ Приел (подпис):………………………….

/проф. *Милен Петров*/