University of West Attica,
Department of Informatics and Computer Engineering,
Winter Semester 2024-2025,
Professor Nikitas N. Karanikolas

1. General Idea / purpose of system

This is your assignment for "**Network Programming**". You have to create one program that services requests about Cities, Businesses, Sights / Monuments, Airports, Hospitals, Museums and Universities. The service program implements a ServerSocket that listens in some port to get requests and provide relevant information (answer the request). Some example requests are:
City Hamburg
City Frankfurt
City Hanover
City Munich
Businesses Hamburg
Sights Frankfurt
Airports Munich
Hospitals Hanover
Museums Berlin
Universities Cologne

2. Data of system

The database of program is two text files with UTF8 encoding. These files are named "cities.txt" and "namedEntities.txt". The format of **"cities.txt"** follows:

Ref#city_la_nom#city_la_acc#city_la_gen#city_la_dat#city_la_loc#city_la_ins#city_la_voc#city_la_gender#city_en_nom#city_state
where:
Ref is the reference number (a unique value),
city_la_nom is the name of city in your language in nominative case, singular,
city_la_acc is the name of city in your language in accusative case, singular,
city_la_gen is the name of city in your language in genitive case, singular,
city_la_dat is the name of city in your language in dative case, singular,
city_la_loc is the name of city in your language in locative case, singular,
city_la_ins is the name of city in your language in instrumental case, singular,
city_la_voc is the name of city in your language in vocative case, singular,
city_en_gender is the gender of the city, a value of the set {'masculine', 'feminine', 'neuter'},
city_en_nom is the name of city in English language in nominative case, singular,
city_state is the state where the city is located, e.g. Bavaria for the city Munich.

Some countries are having other structure for cities. For example Spain has Province instead of State. But it is practically same as the state. So, use the city_state for entering the Province.

If the language is not supporting all these (7) cases, the number of columns are reduced. For example, Modern Greek has only four cases (nominative, genitive, accusative, dative). Also, there are no states in Greece. The following are two examples of cities for Greece/Greek:

Ref	city_gr_nom	city_gr_gen	city_gr_acc	city_gr_dat	city_en_gender	city_en_nom
1	η^Αθήνα	της^Αθήνας	την^Αθήνα	^Αθήνα	feminine	Athens

2	ο^Πειραιάς	του^Πειραιά	τον^Πειραιά	^Πειραιά	masculine	Piraeus

The actual form of these lines (in the text file) will be as following:

1#η^Αθήνα#της^Αθήνας#την^Αθήνα#^Αθήνα#feminine#Athens
2#ο^Πειραιάς#του^Πειραιά#τον^Πειραιά#^Πειραιά#masculine#Piraeus

The symbol ^ is used to separate the inflected form of word from the definite article. If there is not definite article, the symbol ^ is the first character in the field (see column city_gr_dat in the previous table).

"namedEntities.txt" is used to keep information about special names (named entities) for the country. Special names (named entities) can be:

Businesses (Επιχειρήσεις),
Sights / Monuments (Αξιοθέατα / Μνημεία),
Airports (Αεροδρόμεια),
Hospitals (Νοσοκομεία),
Museums (Μουσεία),
Universities (Πανεπιστήμια).

Each line of file "namedEntities.txt" contains information for one special name (named entity). The format of each line (named entity) is:

ent_la_nom#ent_la_acc#ent_la_gen#ent_la_dat#ent_la_loc#ent_la_ins#ent_la_voc#ent_en_gender#city_ref#type
where:
ent_la_nom is the named entity in your language in nominative case, singular,
ent_la_acc is the named entity in your language in accusative case, singular,
ent_la_gen is the named entity in your language in genitive case, singular,
ent_la_dat is the named entity in your language in dative case, singular,
ent_la_loc is the named entity in your language in locative case, singular,
ent_la_ins is the named entity in your language in instrumental case, singular,
ent_la_voc is the named entity in your language in vocative case, singular,
ent_en_gender is the gender of the named entity, a value of the set {'masculine', 'feminine', 'neuter'},
city_ref is a number referring the identifier (ref column of "cities.txt") of the city where the named entity is located,
type is a value from the set {Businesses, Sights, Airports, Hospitals, Museums, Universities}

If the language is not supporting all these (7) cases, the number of columns is reduced. For example, Modern Greek has only four cases (nominative, genitive, accusative, dative). The following are two examples of named entities for Greece/Greek:

ent_gr_nom	ent_gr_gen	ent_gr_acc	ent_gr_dat	ent_en_gender	city_ref	type
η^Ακρόπολη	της^Ακρόπολης	την^Ακρόπολη	^Ακρόπολη	feminine	1	Sights
το^Πανεπιστήμιο Πειραιώς	του^Πανεπιστημίου Πειραιώς	το^Πανεπιστήμιο Πειραιώς	^Πανεπιστήμιο Πειραιώς	neuter	2	Universities

The actual form of these lines (in the text file) will be as following:

η^Ακρόπολη#της^Ακρόπολης#την^Ακρόπολη#^Ακρόπολη#feminine#1#Sights
το^Πανεπιστήμιο Πειραιώς#του^Πανεπιστημίου Πειραιώς#το^Πανεπιστήμιο Πειραιώς#^Πανεπιστήμιο Πειραιώς#neuter#2#Universities

3. Data Structures of system

When the program starts, it loads the data of "cities.txt" in an Arraylist of cityclass classes (name the ArrayList as **ALC**). To do this, the program reads line by line the text file and split

each line (using the split method). The constituents of the split line are used as arguments for the invocation of the cityclass constructor. The new cityclass instance is added to the Arraylist.

In a similar way, the program loads the data of “namedEntities.txt” in an Arraylist of entityclass classes (name the ArrayList as **ALE**).

4. Program responses

Whenever the request is for a city (e.g. “City Hamburg”),

The program tries to find the city (match the input city, e.g. Hamburg with the attribute city_en_nom of an array item from the ALC) and return all attributes of the city, as a string.

Whenever the request is for some type of named entities in a city (e.g. “Airports Munich”),

The program finds the city from the ALC to get the <city identifier> (numeric value of Ref attribute) and next it finds all entries of ALE having the requested type (type=Airports, in our example) and the requested city (city_ref=2, if we assume that Munich has Ref=2). All attributes of each ALE item that matches the requested type and the requested city are returned as a string (one string/line per matching ALE item).

5. General instructions:

Each student should make the work for his/hers country/language. For convenience, I am providing the resources where students can find a list of cities for his/hers country, but only in one case (usually nominative case, singular). As native speakers, students can express the other cases (genitive, accusative, dative, etc) of cities. It is obligatory to elaborate all cities existing in the relevant for the country/language resource.

For the named entities, the expectation is to have one hundred (or more) true named entities. We don’t provide resources for named entities but students can easily find at least one hundred of such from the internet.

In order to have some refreshment about noun declensions and cases, some resources are also provided, for each language of interest. The students can possibly find and use other better resources. All students have to declare the language (noun declensions and cases) resources they have used.

The idea of this program is similar to the idea of SrvSocket1 project (translation service) presented in chapter 8, pages 267-271, of Book “Java for few”. An improved version with Threads (project SrvSocket1wThreads) is able to service many clients simultaneously. It is available later in the same chapter (pages 275-280). The source programs are available in eclass (eclass.uniwa.gr).

Consider that the above programs (Services implementing ServerSocket) are sending Greek text to an existing client (Microsoft Telnet Client) which is an MS-DOS (not windows) application. This is why the first one (SrvSocket1) has the command (line 70) new PrintStream(incoming.getOutputStream(),true,”Cp737”) assuming Greek CodePage 737 (that is Greek for MS-DOS). Similar is line 101 for the improved version (project SrvSocket1wThreads). If you are going to use the MS Telnet Client as yours project client, you have to use in your Service program the correct codepage (instead of ”Cp737”) that supports your language.

Alternatively you can make your own simple GUI client (having a text field for typing the request, a text area box for displaying the response and a client Socket for sending the request and receiving the response). In this case (GUI client) you may need to do in your service (ServerSocket) something like:
new PrintStream(incoming.getOutputStream(),true,”UTF-8”).

The deadline for submitting your project is Friday, January 24, 2025. Together with your project (source code files and data files) you have to include a small documentation (listing the resources [for cities, for noun declensions], few examples of input and output, some screenshots, anything you have to emphasize or make clear). Earlier submissions are preferred in order to have time for improvements.

For assistance find me in my office on this Friday (December 20, 2024) from 12:00 until 14:00. I will stay in the University until 20:00 but I will have lectures from 14:15 until 19:30. Otherwise, find me after January 8th).

The project is separate for each student. If more than one students speaking the same language want to collaborate (work as a team), they have to find me this Friday (December 20, 2024). In such case the requirements will be extended (a little bit).

Resources for cities:
https://en.wikipedia.org/wiki/List_of_cities_in_Germany_by_population
https://www.britannica.com/topic/list-of-cities-and-towns-in-Germany-2038874
https://simplemaps.com/data/de-cities

https://www.britannica.com/topic/list-of-cities-and-towns-in-Spain-2041711
https://en.wikipedia.org/wiki/List_of_metropolitan_areas_in_Spain
https://simplemaps.com/data/es-cities

https://en.wikipedia.org/wiki/List_of_cities_and_towns_in_Moldova
https://simplemaps.com/data/md-cities

https://en.wikipedia.org/wiki/List_of_cities_in_Jordan
https://simple.wikipedia.org/wiki/List_of_cities_in_Jordan
https://simplemaps.com/data/jo-cities

Resources for noun declensions and cases:

https://en.wikipedia.org/wiki/German_declension
https://en.wikipedia.org/wiki/German_nouns
https://www.germanveryeasy.com/noun-declension

https://www.fluentu.com/blog/spanish/spanish-cases/
https://en.wikipedia.org/wiki/Spanish_grammar

https://en.wikipedia.org/wiki/Romanian_nouns

https://www.livelingua.com/peace-corps/Arabic-Jordanian/Jordanian%20Arabic%20Grammar%20for%20Beginners.pdf

Using split method for a String object

Java code
```
String tmp="1#η^Αθήνα#της^Αθήνας#την^Αθήνα#^Αθήνα#feminine#Athens";
String tokens[] = tmp.split("#");
for (int i=0; i<tokens.length; i++)
   System.out.printf("%2d. %s\n", i+1, tokens[i]);
```

Output
```
 1. 1
 2. η^Αθήνα
```

3. της^Αθήνας
4. την^Αθήνα
5. ^Αθήνα
6. feminine
7. Athens

Professor Nikitas N. Karanikolas,
Department of Informatics and Computer Engineering,
University of West Attica,
Tel. +30-210-5385737,
Fax +30-210-5910975,
nnk@uniwa.gr
(Uni.W.A. web page EN) http://www.ice.uniwa.gr/en/emd_person/nikitas-karanikolas/
(Uni.W.A. web page EL) http://www.ice.uniwa.gr/emd_person/17335/
(personal web page) http://users.uniwa.gr/nnk/
(dblp web page) http://dblp.uni-trier.de/db/indices/a-tree/k/Karanikolas:Nikitas_N=.html
(scopus web page) http://www.scopus.com/authid/detail.url?authorId=6507018215
(Google Scholar profile) https://scholar.google.com/citations?hl=en&user=FREbt6sAAAAJ