**SyntheaTM Patient Generator [Enhancement]**

SyntheaTM is a Synthetic Patient Population Simulator. The goal is to output synthetic, realistic (but not real), patient data and associated health records in a variety of formats.

In addition to default implementation, this enhancement version includes the following:

* Modified Disease Module (Condition), to generate ICD-10 conditions.
* Generate data in alternative geographic locations with relevant geographical standard, such as Europe.
* Generate Questionnaires, QuestionnairesResponses type of FHIR R4 resources in a flexible and configurable way.

### **Generate Synthetic Patients:**

Generating the population one at a time...

./run\_synthea

By default, it will include one questionnaire. Command-line arguments need to be provided to specify a number for questionnaires to be loaded. The argument “-qr” specifies the number of questionnaires to be loaded.

Usage is:

run\_synthea [-s seed] [-p populationSize] [-m moduleFilter] [state [city]] [-qr questionnairesCount]

For example:

run\_synthea -qr 7

**Disease Modules:**

To generate ICD-10 conditions it’s required to have the modules specify the correct ICD-10 codes for the specified diseases. By default, Synthea contains SNOMED-CT codes for the diseases in its disease module json files. So, to have ICD-10 conditions, the following steps needs to be performed in order:

* Take a single disease JSON file and find a single Condition node containing SNOMED-CT code.
* Note down the SNOMED-CT code and the disease description from the JSON.
* Find out the ICD-10 disease from the ICD-10 WHO database (available at: https://icd.who.int/browse10/2019/en) corresponding to the noted SNOMED-CT disease
* Replace the SNOMED-CT code and description in the JSON with the ICD-10 code and description.
* Repeat this for all the Conditions in all the disease JSON files.

**Data from alternative geographic locations:**

By default, Synthea supports generation of patients only based in the US region. However, it’s possible to generate patients from other regions as well just by changing some of its configuration files. In this step, the geography data, providers data as well as the payers’ data need to be configured. This is a huge process, but it is greatly simplified by using an already available dataset from the official synthea git. The synthea-international git repository (https://github.com/synthetichealth/synthea-international) contains the dataset for generating data for international regions e.g., Great Britain, Germany, France etc.

To use this repository, the following steps needs to be performed in order:

* Clone the synthea-international repository alongside the synthea repository.
* Decide on a specific region to use. (for example: Germany – de)
* Copy all the files from the regional subdirectory (in our case ‘de’ subdirectory) to the root of the synthea repository

These steps are enough to generate patient data corresponding to the new region. But this requires we provide a state explicitly as a parameter when calling the run\_synthea command. However, it fails with an exception when run without a state parameter. This happens because of the default state defined in the code is still one from the US region.

To fix that extra steps needed to be performed:

* Define the default state to be fetched from the configuration file (synthea.properties) in Generator.java

`public static String DEFAULT\_STATE = Config.get("generate.default\_state", "Massachusetts");`

* In the configuration file (synthea.properties), define the default state as one from our target region (a German state for our example: ‘Nordrhein-westfalen’)

This will generate sythea data without passing the state parameter explicitly in the command line.

**Code Lifecycle:**

|  |  |  |
| --- | --- | --- |
| Load Quesionnaires from csv (i.e. stored in the location “src\main\resources\providers”) | Generate QuesionnairesResponses | Add QuesionnairesResponses to FhirR4 bundle data |
| Class: Provider.java  Method: csvLineToQuestionnaire()  Class: Provider.java  Method: loadQuestionnaire()  Class: Generator.java  Method: main()  Class: App.java  Method: main() | Class: Provider.java  Method: getQuestionnaireresponse()  Class: Generator.java  Method: updatePerson()  Class: EncounterModule.java  Method: createEncounter()  Class: EncounterModule.java  Method: process()  Class: Generator.java  Method: createPerson()  Class: Generator.java  Method: generatePerson() | Class: FhirR4.java  Method: convertToFHIR()  Class: FhirR4.java  Method: questionnaireResponse() |