XText Approach:

In the XText approach, we define the grammar for our controlled natural language using XText. XText generates a parser and other necessary language infrastructure code for us. It also generates a file called JvmModelInferrer which is responsible for generation of java code corresponding to the query.

Database:

dataDB : This mysql database consists of two tables :

- **region_details:** having information about various morphological parameters like regionID, buildings, treesArea, bushesArea, lakeArea for all the regions
- **LCZ_LST:** having mapping of various LCZs to minTemp, maxTemp and avgTemp

Steps Involved:

For Data simulation:

1. The implementation of the data simulator is done using '*Datasimulation*' class.
2. It has different methods for each of the 17 LCZs. Methods RandomLCZ_1() to RandmoLCZ_10() for each urban LCZ and methods RandomLCZ_A() to RandomLCZ_G() for rest of the LCZs. Each method is used to generate entries for that particular LCZ.

For grammar:

1. First XText is to be installed as a plugin in eclipse. The steps of installation are given below in the 'Tools Used' section.
2. After installing XText, create a new XText project in eclipse - File -> New -> Project -> XText Project. Define the *main project name, language name*, *DSL-File extension* according to your choice.
3. A total of 5 new projects are created. In the first project, find a file called '*Extension.xtext*' where *Extension* is defined in the above step in the src folder. In this file we define the grammar for our DSL.
   Note: In the default .xtext file, first line is -
   grammar org..xtext.Mycnl with org.eclipse.xtext.common.Terminals
   We replace it with -
   grammar org.xtext.Mycnl with org.eclipse.xtext.xbase.Xbase
4. After defining the grammar, we generate xtext artifacts by choosing Run As -> Generate Xtext Artifacts from the context menu of the grammar editor.
5. Now a file named JvmModelInferrer.xtend is generated in the folder named org/xtext/jvmmodel in the src folder. This file is used for generation of java code corresponding to the query.
6. After defining how to generate java code in the jvmmodelInferrer file, we launch a new eclipse instance by clicking green button in the toolbar or by Run As -> Eclipse Application. This creates a new eclipse window.

7. In the new window, create a Java project (File -> New -> Project... ->Java Project). In this project, we create a new file with the extension we chose in the beginning in the src folder of java project.
8. In this file we write our queries in DSL. In the src-gen folder, corresponding java files for our queries are generated. Also, extra supplementary classes like region.java, building.java and LCZ_LSTMapper.java are created. The file 'DetailsExtractor.java' is executed to give the desired results.

Tools Used:

XText: Xtext is an open-source software framework for developing programming languages and domain-specific languages. Unlike standard parser generators, Xtext generates not only a parser, but also a class model for the abstract syntax tree, as well as providing a fully featured, customizable Eclipse-based IDE.

1. Installing XText:1. Choose Help -> Install New Software... from the menu bar and Add....
2. Insert one of the update site URLs below
    a. https://download.eclipse.org/modeling/tmf/xtext/updates/composite/releases/
    b. https://download.eclipse.org/modeling/tmf/xtext/updates/composite/latest/
    c. https://download.eclipse.org/modeling/tmf/xtext/updates/composite/milestone/
3. Select the Xtext SDK from the category Xtext and complete the wizard by clicking the Next button until you can click Finish.
4. After a quick download and a restart of Eclipse, Xtext is ready to use.