Lab Assignment 2: **HPO Explorer Tool**

Aim – **To implement a simple software prototype tool, named HPOExplorer, to explore a biomedical ontology of human disease description terms, stored in a structured text file, in response to a simple type of queries about these terms. HPOExplorer receives these queries from, and produces results to, formatted text files.**

The two main objectives in this assignment are: 1) to implement simple algorithms to parse and traverse a set of structured data, that is an ontology description stored in a text file, with the objective of assembling structured responses to simple queries about human disease classification and definitions, and 2) to give you practice applying the file I/O and exception classes in the Java API, as well as designing and coding simple classes that will allow you to use sets of objects to implement the capabilities of this ontology exploration tool.

To attain these goals, it is advisable that you at least design and implement the following Java classes:

1) A class **Term** to keep track of each of the concepts (i.e., terms) defined in the ontology.
2) A class **Query** to store input queries and their results.
3) A main driver class **HPOExplorer** to: 1) read the ontology's text file, parse it and build the data structures that support exploring it, 2) read a set of queries from a text file and produce the applicable results in another formatted text file, and 3) produce a text file displaying the results of a general query on the entire ontology.

General Description

As a software developer supporting a research team at a biomedical institution, your task is to write a Java program, named HPOExplorer, which would allow someone to query the publicly available HPO (Human Phenotype Ontology) ontology.

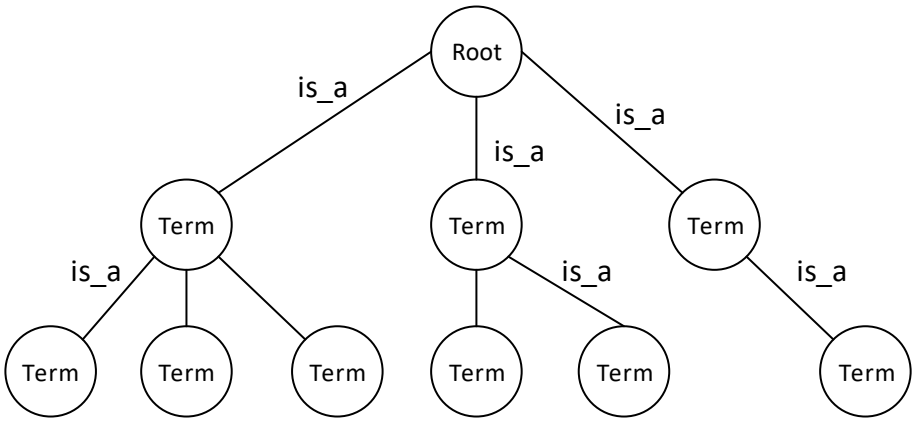
In computer science, an **ontology** is a formal representation of the knowledge by a set of concepts within a domain and the relationships between those concepts. It is used to reason about the properties of that domain and may be used to describe the domain.

The **HPO ontology** (provided to you in file **HPO.txt**) contains a formal representation of the knowledge required to describe the manifestation of diseases in the different human phenotypes (for more information check this link https://hpo.jax.org/app/). In simple terms, for the purpose of this assignment, **a human phenotype** is the sum of the genetic influence factors, observable physical traits and identifiable environmental interactions that define and make a group of individuals unique and different from other groups.

File HPO.txt is organized as follows:

- Each line of the file either starts with a **keyword** (e.g., ontology, def, is_a, etc.) followed by a ":" (colon), or it contains just the text "[Term]", or it is empty, and is terminated with CR (Carriage Return) and LF (Line Feed).
- File HPO.txt is organized in **sections** separated by a blank line.

- **The initial section** of the file contains just a **manifest** storing information about versions, implementors, contributors and references to sources of information used to develop the HPO. This section should be skipped in your analysis of the HPO.txt file.
- After the initial section, the file contains 13,941 sections starting with the "**[Term]**" tag in a single line. Each of these sections defines and/or describes **a concept** and its **relationship**(s) to other concepts in the HPO ontology. It is composed of a set of text lines starting with one of the following keywords: **alt_id**, **comment**, **consider, created_by**, **creation_date**, **def**, **id**, **is_a**, **is_anonymous**, **is_obsolete**, **name**, **property_value**, **replaced_by**, **subset**, **synonym**, **xref**.
- The term sections in HPO are organized as an acyclic graph (tree with no circular paths) of nodes and links, having a root node (see Fig. 1). The nodes represent a term section identified by the keyword **id**. The links represent a parent-child relationship between two terms and are implemented by an id value associated with the **is_a** keyword. The root node is **id**: HP:0000001. A parent node can have multiple children nodes associated with it. Sometimes, a node, other than the root node, can have multiple alternative parent nodes associated with it. This situation can be treated as if a node can belong to multiple trees at the same time. However, for traversal purposes, only one tree is considered.



- Fig. 1 -

- Moving up from the bottom of the ontology tree in the direction of the root, generically means classifying terms into more general categories of concepts.
- Moving down from the root node to the bottom of the tree, generically means describing in more detail the elements of categories of concepts.

In this assignment, we will only consider queries where the **id** fields of single terms other than the root, located anywhere in the tree, are provided as arguments. The queries should return in every case, all the information about the nodes themselves, along with all the information about all the nodes forming a path to the root node.

Program Implementation and Others

I – Ontology's tree construction and traversal

The Java program that you will implement should:

- Read all the information of all the term nodes (a total of 13,941) from file **HPO.txt** file into a data structure that emulates the tree of relationships between them (see Fig. 1).
- Read a **queries.txt** file, provided to you, containing a set of single-line queries. Each line of this file starts with the keyword **query** followed by ":", a space, the string "HP:", and a seven-digit number indicating the **id** of a term in the HPO.txt file.
- Generate a **results.txt** file containing the answer to each of the queries provided in the **queries.txt** file. Each answer consists of the term sections for all the nodes, in a path from the node indicated in the query (including it) all the way to the root node. As in the HPO.txt file, the term sections are separated by a blank line. The answer to each query begins with a **[query_answer]** line and ends with a blank line.
- Produce a **maxpath.txt** file containing all the term sections corresponding to the **longest path** from a node to the root node. As in the results.txt file, the term sections in this path are separated by a blank line. The first line of this file is the string **[max_path=length]**, where **length** is the maximum number of links to reach the root node from any node.

II – Submission

Compress the source code for classes **HPOExplorer, Term** and **Query**, as well as files **results.txt** and **maxpath.txt**, into a single zip archive file named:

[your student number]_Assignment2.zip

For example, if your student number is 12345678 then you will name your submission file as 12345678_Assignment2.zip

Submit file **[your student number]_Assignment2.zip** via onQ by **February 25, 2018, 11:00 p.m.**

III – Marking Scheme

• Implementation of class **HPOExplorer**	8 points
• Implementation of classes **Term** and **Query**	4 points
• Generation of **results.txt** file	3 points
• Generation of **maxpath.txt** file	3 points
• Code style and documentation	2 points
TOTAL:	20 points