ABSTRACT

FINDING ROLE ERRORS OF THE NCIT GENE HIERARCHY USING THE NCBI

by
Marc Oren

Due to the knowledge discovery process of gene information, details such as organism and chromosomal location should be known. Furthermore, the extensive biomedical research in genomics led to discovery of processes and diseases in which a gene plays a role. In the Gene hierarchy of the NCI Thesaurus (NCIT) such knowledge is represented by appropriate roles. However, upon review of the Gene hierarchy of the NCIT, many role errors are found. Realizing that such details are provided by another knowledge repository of NIH, the NCBI gene database, a methodology is presented to use NCBI to discover role errors for the Gene hierarchy.

For this, a web crawler was developed to retrieve the knowledge from the NCBI, so it could be represented in a compatible way with the NCIT gene roles, to facilitate comparison. The most difficult challenge is with process roles for which one gene is playing role in several processes and thus several targets exist for one gene. A procedure is developed to explore process role errors in the Gene hierarchy of the NCIT utilizing the Biological Process hierarchy of the NCIT and NCBI gene database.

FINDING ROLE ERRORS OF THE NCIT GENE HIERARCHY USING THE NCBI

by
Marc Oren

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

Department of Computer Science

May 2006

APPROVAL PAGE

FINDING ROLE ERRORS OF THE NCIT GENE HIERARCHY USING THE NCBI

Marc Oren

Dr. Barry Cohen, Thesis Advisor — Date
Assistant Professor of Computer Science, NJIT

Dr. Yehoshua Perl, Committee Member — Date
Professor of Computer Science, NJIT

Dr. Michael Halper, Committee Member — Date
Professor of Computer Science, Kean University

BIOGRAPHICAL SKETCH

Author: Marc Oren

Degree: Master of Science

Date: May 2006

Undergraduate and Graduate Education:

- Master of Science in Computer Science,
  New Jersey Institute of Technology, 2006

- Bachelor of Arts in History,
  University of Michigan, Ann Arbor, 1998

Major: Computer Science

To my beloved family

ACKNOWLEDGMENT

I would like to express my deepest appreciation to Dr. Barry Cohen, who served as my research supervisor. Special thanks are given to Dr. Yehoshua Perl for actively participating in my committee and constantly pushing me to strive for more. Thanks are also due to Dr. Michael Halper for serving on my committee.

Special thanks to Hua Min for her assistance throughout our research.

As well, I greatly appreciate my wife's support as she stood by me throughout this arduous process.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Objectives

Recent studies show that some biomedical vocabularies do not conform to principles of good practice in terminological and ontological design [1, 2, 3, 4]. Due to the size, complexity and rapid development of such terminologies, it is unavoidable that errors are introduced into them. Consequently, auditing and correction of large terminologies is a major challenge facing the biomedical informatics community.

Genomic research is at the forefront of bioscience. Gene terminology plays a critical role for genomic research. The rapid growth of genomic information over the past few years and the computationally inferred nature of much of genomic knowledge make it particularly susceptible to error. Thus it is important to provide an efficient methodology of quality assurance for the genomic components of medical terminologies. In this work, a structural auditing methodology [5] was applied to audit two hierarchies of the National Cancer Institute Thesaurus (NCIT). The following presents the results of auditing of role errors in the gene hierarchy of the NCIT.

The gene database of the National Center for Biotechnology Information (NCBI) is used as a comparative source for the auditing.

1.2 Background

The National Cancer Institute Thesaurus (NCIT) is a controlled terminology that provides broad coverage of the cancer domain. Its topics include diseases, drugs, chemicals, diagnoses, genes, anatomy, organisms, and proteins, etc. [6]. The NCIT is a public domain terminology that follows a description logic based model [7, 8, 9].

The NCIT data model has four basic components: Concepts, Kinds, Roles and Properties. The basic unit of knowledge in the NCIT is the concept. The NCIT (2005 release) contains approximately 42,000 concepts partitioned into 21 disjoint hierarchies such as Biological Process, Gene, Gene Product, etc. Each hierarchy consists of IS-A relationships between child and parent concepts forming a directed acyclic graph (DAG). Roles are directed edges defining relationships between concepts. Lateral (non-hierarchical) relationships among concepts are referred to as associative or semantic roles in contrast to the (hierarchical) IS-A relationships. All roles are passed from a parent concept to a child concept in an inheritance hierarchy. Kinds are disjoint sets of concepts and represent major subdivisions of the NCIT.

Each concept belongs to only one kind. Kinds are used in role definition to constrain the domain and range values for that role. For example, the domain of the role "gene plays role in process" is restricted to the Gene Kind and its range is restricted to the Biological Process Kind.

Properties describe a concept. Examples include: definition, preferred name, synonym, and semantic type.

CHAPTER 2

STRUCTURAL CHARACTERISTICS OF THE NCIT HIERARCHY

2.1 Problem Statement

The Genome hierarchy of the National Cancer Institute Thesaurus (NCIT) will be audited for areas containing large errors. The NCIT contains healthcare terminologies currently being used in over 40 countries. Auditing is an intensive effort requiring knowledge of a domain. Manual auditing will be too labor intensive to allow for iteration of the process. Automation will be attempted to find these errors.

2.2 NCIT Concepts

There are 1,786 concepts in the Gene hierarchy of the NCIT (2004 version). There are 1,554 concepts located at the leaves of this hierarchy, i.e., these concepts have no children. They are the actual gene concepts.

2.3 Gene Subset

The 232 internal concepts serve to classify the genes into categories. The Gene hierarchy is different from other NCIT hierarchies in that the internal concepts are not gene concepts, just categories of genes. In contrast, an internal concept of the Biological Process hierarchy can be a process with more processes as children.

For example, the Cancer Progression internal concept describes a process, and has 12 descendants. There are only 42 concepts with two parents, and all are gene concepts. Examples include GRB7 Gene, MADD Gene, and MAGED1 Gene.

2.4 Importance of Auditing the Gene Hierarchy

One of the fastest expanding areas of biomedical research concerns the knowledge of genes and genomes. The Human Genome Project (HGP) obtained a near or nearly complete reading of the 3.2 billion nucleotides comprising the human genome and computationally identified more than 20,000 human genes. Obtaining a comprehensive human genome sequence has strongly impacted areas of biomedical research and medicine [10].

For example, the identification of variations (alleles) of a gene may permit the development of diagnostic tests that reveal potential health problems before they manifest themselves as symptoms [11]. Knowing a patient's genetic makeup may allow physicians to minimize certain disease risks [12]. The results produced by the genome project enhance our understanding of human heredity.

2.5 Gene Ontology

The Gene Ontology (GO) [13] provides a controlled terminology allowing researchers to report their results regarding genes and gene products. It continues to be an important resource in the molecular biology domain. Many model organism databases devote substantial resources to annotating the genes in their databases with GO codes. GO is composed of three disjoint components: cellular components, molecular functions, and biological processes. As of December 2005, GO contained 1,681 component terms, 7,384 function terms, and 10,291 process terms.

The UMLS integration effort involving GO's concepts is reported in [14]. However, according to the design policy of the UMLS, only concepts and relationships from GO were incorporated. No relationships to the rest of the UMLS were added. The NCIT has added such relationships.

CHAPTER 3

IMPLEMENTATION

3.1 Auditing Techniques

In general, an auditing technique should minimize the effort of the auditor while maximizing the likelihood of finding mistakes. In this work, an efficient technique for auditing large terminologies based on partitions and their derived associated abstractions is demonstrated. This process takes a subset of a terminology and researches it for validity. Forming a smaller group of similar concepts facilitates the identification of missing roles to a concept, those which would naturally be expected to belong to the group but are presently absent. Such situations could arise because the roles were omitted from the terminology originally or because they are misclassified or misplaced in the hierarchy.

3.2 Previous Work

In previous work, the importance of semantic uniformity and cohesiveness of the groups of a partition was investigated. Though the areas were found to be structurally uniform, many of them were found to lack semantic uniformity. Particularly, areas lack unique roots. The root is the general ancestor of all other concepts in that specific area. Some forms of partitioning have been carried out by terminology designers and are included as inherent aspects of terminologies. For example, SNOMED and NCIT have 18 and 22 top-level disjoint hierarchies, respectively. This is meant to capture high-level subject

areas. Overall, such hierarchies and their sizes provide a picture of a terminology's range and depth of knowledge.

3.3 Design

Programs were created for retrieving data from the NCIT and NCBI databases via their web interfaces. The data retrieved is stored in a database that contains multiple tables designed to limit replication and promote efficient searching. An interface to search the data and make logical associations for data between the NCIT and NCBI is created. Another set of programs reads the initial set of input data and write a delimited report to open in Microsoft Excel. To compare the data from the 2 databases (NCIT and NCBI), the data differences are displayed in Microsoft Excel with differences highlighted in the specific areas researched.

3.4 Input Files

Two Java programs were written to collect and analyze the data. These programs have three input files.

3.4.1 Configuration File

The first input file is a configuration file. The java.util.properties class [15] is used to load and read all of the configurable parameters of the web crawler. Configurable parameters are detailed in the subsequent table.

3.4.2 Configuration File Parameters

The following table details the configuration file parameters created and used.

Table 3.1 Configuration File Parameters

Category	Property Name	Meaning of Value
Debugging	Debug	To write out detailed informational messages as programs run
Debugging	writeOutput	Write output to files? (Y/N)
Input	GenesFile	Input file detailing names and concept ID's of genes
Input	synonymFile	Input file containing synonyms of all concepts in biological process hierarchy
Output	printNCBI	Print NCBI data
Output	printNCI	Print NCI data
Output	FinalOutputFileName	File name of output file to which date will be appended for a single output file
Output	geneOutputFileName	Output file for recommendations for gene modifications
Output	chromOutputFileName	Output file for chromosomes
Output	processOutputFileName	Output file for processes found per gene for each data source
HTTP	proxyHost	IP address of proxy server (blank if none)
HTTP	proxyPort	Port proxy server listens (blank if none)
HTTP	ncbiBaseURL	Domain name with protocol of NCBI server
HTTP	NCBIQuery	Completion of query string for NCBI application
HTTP	nciBaseURL	Domain name with protocol of NCI server

3.4.3 Synonym File

The second input file represents all synonyms found for processes within the biological process hierarchy of the NCIT. This data is used to determine if a process found in the NCBI is actually a synonym of an existing process in the NCIT.

To create this file, all process concepts in NCIT are collected. These represent all the descendants of the biological process concept. When loaded, this file is structured into an indexed array.

3.4.4 Genes File

The third input file is a list of genes, documenting the subset of genes within the scope of this process.

3.5 Output Files

The programs have three output files.

3.5.1 Raw Output File

This file contains all data collected in comma-delimited format. See Appendix C.1 for a hyperlink to the file.

3.5.2 Gene Output File

The gene file contains a list of every gene with the recommended action (if any) to be taken on the process associated with that gene to correct a possible error.

Possible actions:

1. Addition of existing concept
2. Addition of synonym to concept
3. Addition of new concept to BP and as gene role target
4. No action, same concepts found in both databases
5. No action, a more specific concept found in NCIT
6. No action, synonym concept found in NCIT
7. Replace existing concept with child concept
8. Replace concept with synonym of found concept

See Appendix B.1 for a hyperlink to the file.

3.5.3 Chromosome Output File

The chromosome file contains a list of every gene researched with the chromosome location listed in the NCBI database and the NCIT database.

See Appendix A.1 for a hyperlink to the file.

3.6 Programming

To collect the data, a series of web crawlers were developed. An http interface is used to communicate with the two web interfaces provided by the NCBI and the NCIT.

3.6.1 Data Collection Program

The first program written utilizes the Java method java.net.URLConnection [15]. Using the GET method, the web page is requested via the URL posted to the appropriate website. The NCIT or NCBI response is then parsed for the gene name, the organism associated with the gene (Homo sapiens), the location where the gene is found, and the processes associated with this gene. If a link is presented, the link is followed and the data parsed. The code for this program is detailed in Appendix E.

3.6.2 Data Analysis Program

The second program reads the output of the program described in sub-Section 3.6.1. It then compares the information found and writes the output files listed in sub-Sections 3.5.1, 3.5.2, and 3.5.3.

An algorithm was developed for this analysis. This algorithm is detailed in Appendix D and the code is detailed in Appendix F.

CHAPTER 4

CONCLUSION

4.1 Results

This project sought to identify improvements to the NCIT thesaurus by comparing it to a second genomic data source, the NCBI. A number of automated methods were developed to collect this data, compare the two data sets, identify possible omissions, misclassifications and other errors and to recommend corrective actions, based upon a divide and conquer auditing technique. This allowed for thorough and comprehensive auditing methodology. From this work, two major areas of improvement were found in the NCIT thesaurus.

4.2 Chromosome Locations

There is a large number of differences between the chromosome locations listed in NCIT and NCBI. Of the 1,960 genes researched, 576 had different locations in NCBI and NCIT – 22.71%.

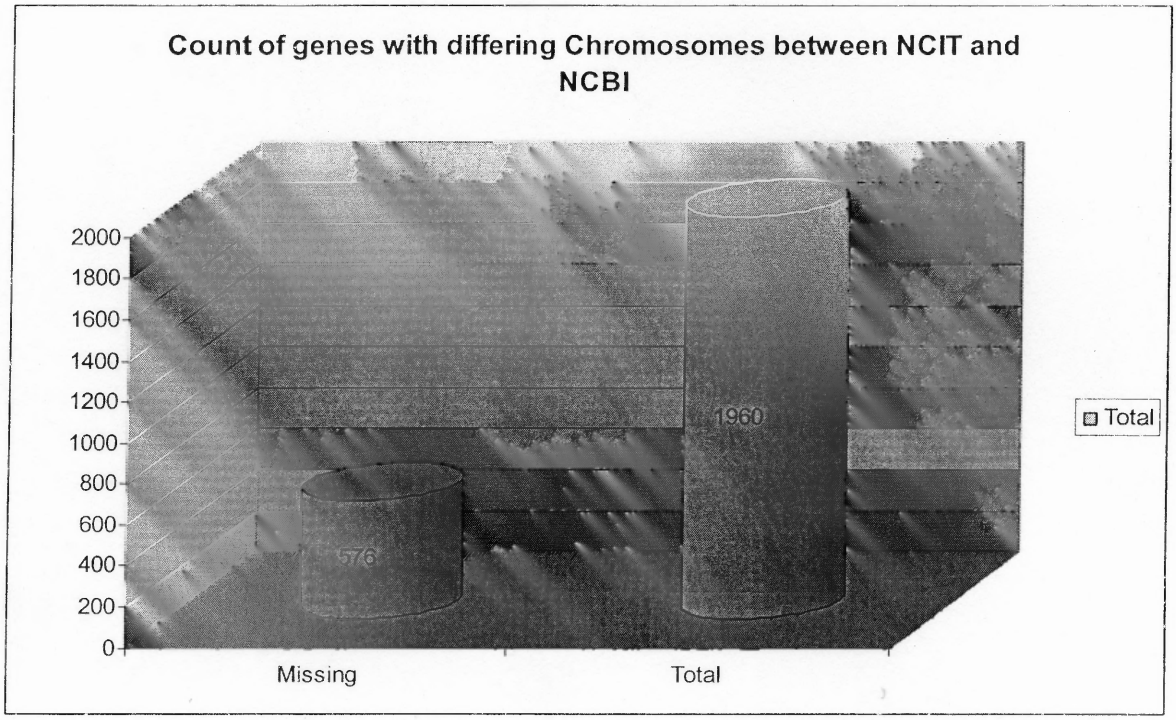

Figure 4.1 Number of genes with a difference in chromosome location between NCIT and NCBI.

4.3 Processes: Additions and Modifications

Processes associated with genes were found to have many differences. In total, 6,125 process role instances were found associated with the 1,970 gene concepts. Of these, 1,355 processes are recommended for addition to genes as known processes in the biological hierarchy. The number of processes recommended for addition to both the biological process hierarchy and as process role's targets for specific genes is 3,824. The number of processes found to have synonym concepts requiring addition to the NCIT's gene is 300 (4.9%). The number of processes found in both databases for the same gene is 584 (9.5%). The fraction of concepts found in both databases where the NCBI provided a synonym concept to the process known in the NCIT thesaurus is 0.7%.

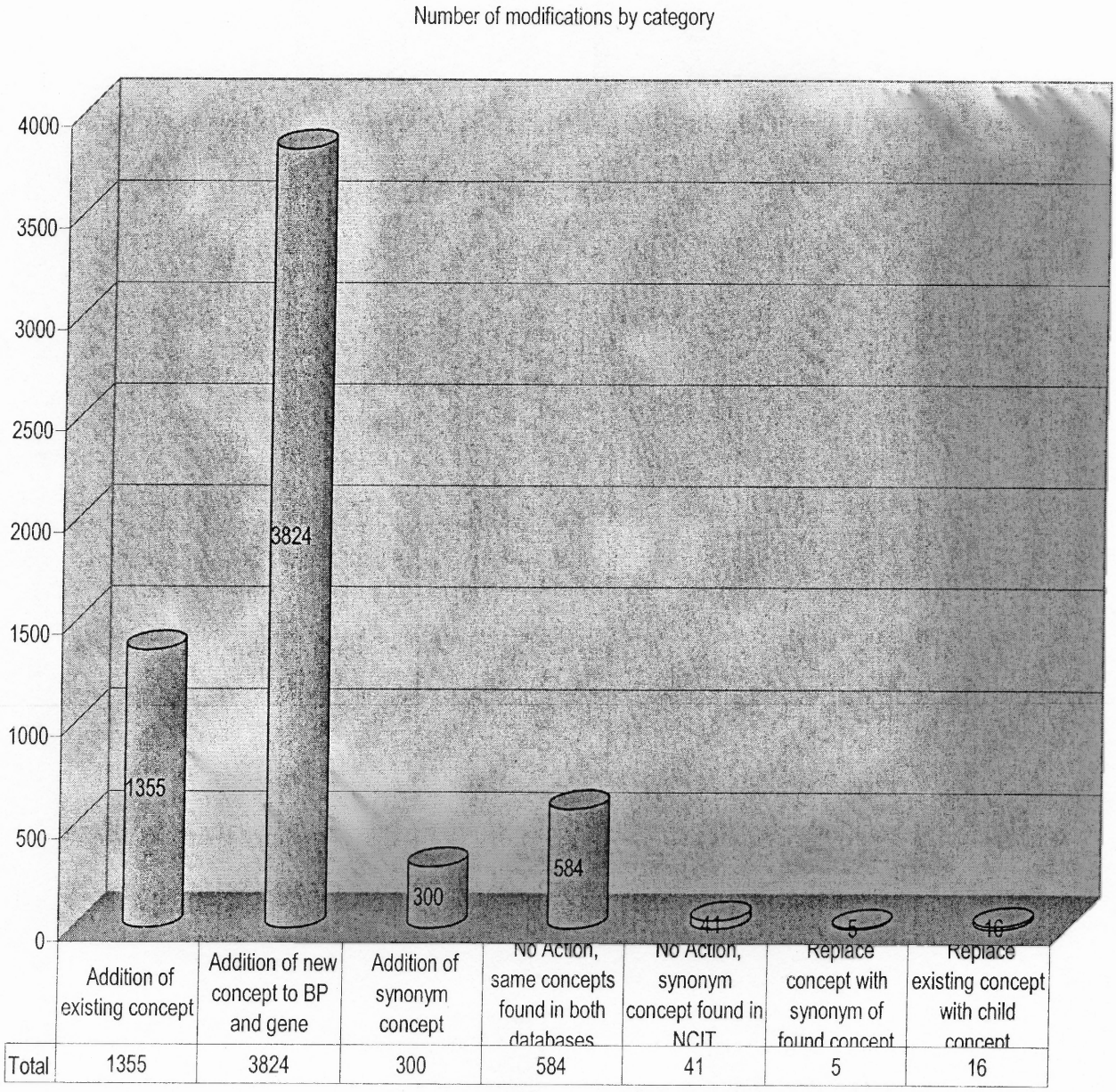

Figure 4.2 Detailed numbers of recommended modifications.

APPENDIX A

CHROMOSOME DATA OUTPUT FILE

The complete chromosome data output file can be found at the following web location:

http://web.njit.edu/~mo24/thesis [16]

Table A.1 Sample of Chromosome Data Output File

Gene Name	NCBI Chromosome Location	NCI Chromosome Location
ereg_gene	4q13.3	4q21.1
top1_gene	20q12-q13.1	
sui1_gene		17q21.2
irak2_gene	3p25.3	3p25.3

APPENDIX B

GENE DATA OUTPUT FILE

The complete gene data output file can be found at the following web location:

http://web.njit.edu/~mo24/thesis [16]

Table B.1 Sample of Gene Data Output File

Gene Name	NCBI Process	Status	NCI Process
tf_gene	ion transport	Addition of new concept to BP and gene	
tf_gene	iron ion homeostasis	Addition of new concept to BP and gene	
tf_gene	iron ion transport	Addition of new concept to BP and gene	
adamts1_gene	proteolysis	No Action, same concepts found in both databases	proteolysis
adamts1_gene	negative regulation of cell proliferation	Addition of synonym concept	
adamts1_gene	integrin-mediated signaling pathway	Addition of new concept to BP and gene	
plg_gene	induction of apoptosis	Addition of existing concept	
plg_gene	proteolysis	Addition of existing concept	
plg_gene	blood coagulation	Addition of synonym concept	
plg_gene	negative regulation of cell proliferation	Addition of synonym concept	

APPENDIX C

RAW DATA OUTPUT FILE

The complete raw data output file can be found at the following web location:

http://web.njit.edu/~mo24/thesis [16]

Table C.1 Sample of Raw Data Output File

NCBI Gene Name	NCBI Chromosome Location	NCBI Process list	NCI Chromosome Location	NCI Process list
tf_gene	3q22.1	ion transport,iron ion homeostasis,iron ion transport,	3q22.1	ligand binding,transport process,metal ion binding,immune function,
bcl7a_gene	12q24.13		12q24.13	
tbp-associated_factor_gene				transcriptional regulation,transcription initiation,transcription,
hydrolase_gene				hydrolysis,
adamts1_gene	21q21.2	integrin-mediated signaling pathway,negative regulation of cell proliferation,proteolysis,		angiogenic inhibition,inhibition of cell proliferation,proteolysis,
psmd10_gene	xq22.3		xq22.3	ubiquitinated protein degradation,

NCBI Gene Name	NCBI Chromosome Location	NCBI Process list	NCI Chromosome Location	NCI Process list
plg_gene	6q26	blood coagulation,fibrinolysis,induction of apoptosis,negative regulation of angiogenesis,negative regulation of blood vessel endothelial cell migration,negative regulation of cell proliferation,organismal physiological process,proteolysis,tissue development,	6q26	angiogenesis,proteolytic processing,metastasis suppression,
phlda2_gene	11p15.5		11p15.5	
fgfr1_gene	8p11.2-p11.1	mapkkk cascade,cell growth,fibroblast growth factor receptor signaling pathway,protein amino acid phosphorylation,protein amino acid phosphorylation,skeletal development,	8p11.2-p11.1	cell differentiation,receptor signaling,ligand binding,myeloproliferation,tyrosine phosphorylation,cell proliferation regulation,
tal1_gene	1p32	cell differentiation,cell proliferation,regulation of transcription,dna-dependent,	1p32	transcriptional activation,dna binding,hematopoiesis,

NCBI Gene Name	NCBI Chromosome Location	NCBI Process list	NCI Chromosome Location	NCI Process list
kras2_gene			12p12.1	signal transduction,
rbm6_gene	3p21.3	rna processing,	3p21.3	tumor suppression,ligand binding,rna binding,
cd14_gene	5q31.1	apoptosis,cell surface receptor linked signal transduction,inflammatory response,phagocytosis,	5q31.1	receptor signaling,cell adhesion,ligand binding,apoptosis,
mas1_gene	6q25.3-q26	g-protein coupled receptor protein signaling pathway,cell proliferation,morphogenesis,signal transduction,	6q24-q27	cell proliferation,g protein-coupled receptor signaling,ligand binding,
protease_gene				proteolysis,
plau_gene	10q24	blood coagulation,chemotaxis,fibrinolysis,proteolysis,proteolysis,signal transduction,		proteolysis,anticoagulation,

NCBI Gene Name	NCBI Chromosome Location	NCBI Process list	NCI Chromosome Location	NCI Process list
igf2_gene	11p15.5	cell proliferation,development,imprinting,insulin receptor signaling pathway,physiological process,regulation of progression through cell cycle,skeletal development,	11p15.5	stimulation of cell proliferation,intercellular communication,
pard6a_gene	16q22.1	cell cycle,cell division,establishment of cell polarity,intercellular junction maintenance,viral life cycle,	16q22.1	signal transduction,cell proliferation,ligand binding,
gnaz_gene	22q11.22	g-protein coupled receptor protein signaling pathway,signal transduction,	22q11.22	signal transduction,
smarcc2_gene	12q13-q14	chromatin assembly or disassembly,chromatin remodeling,negative regulation of transcription,dna-dependent,positive regulation of transcription,dna-dependent,regulation of transcription from rna polymerase ii promoter,	12q13-q14	chromatin remodeling,transcriptional regulation,protein-protein interaction,

NCBI Gene Name	NCBI Chromosome Location	NCBI Process list	NCI Chromosome Location	NCI Process list
collagen_gene				wound repair,organogenesis,cell movement,cell adhesion,cell growth,morphogenesis,
ddx1_gene	2p24	development,glycolysis,regulation of translational initiation,ribosome biogenesis,spliceosome assembly,	2p24	
oncogene_yes-1				signal transduction,cell growth,cell differentiation,cell proliferation,phosphorylation,tyrosine phosphorylation,
slc5a5_gene	19p13.2-p12	ion transport,sodium ion transport,	19p13.2-p12	drug efflux,transport process,multidrug resistance,ligand binding,

NCBI Gene Name	NCBI Chromosome Location	NCBI Process list	NCI Chromosome Location	NCI Process list
ahr_gene	7p15	apoptosis,cell cycle,regulation of transcription,dna-dependent,response to stress,response to xenobiotic stimulus,signal transduction,transcription from rna polymerase ii promoter,	7p15	receptor signaling,xenobiotic metabolism,transcriptional regulation, cell migration,embryogenesis,cell proliferation,pattern formation,cell fate control,signal transduction,intercellular communication,
wnt5a_gene	3p21-p14	cell-cell signaling,frizzled-2 signaling pathway,morphogenesis,signal transduction,	3p21-p14	
extracellular_matrix_protein_gene				organogenesis,wound repair,morphogenesis,cell movement,cell adhesion,cell growth,
rsu1_gene	10p13	signal transduction,	10p13	signal transduction,

NCBI Gene Name	NCBI Chromosome Location	NCBI Process list	NCI Chromosome Location	NCI Process list
serpinb5_gene	18q21.3	cell motility,	18q21.3	cell-matrix adhesion,ligand binding,metastasis suppression,tumor suppression,
ny-br-20_gene			15q23	
selp_gene	1q22-q25	cell adhesion,cell adhesion,	1q22-q25	cell-matrix adhesion,immune function,ligand binding,
birc6_gene	2p22-p21	anti-apoptosis,apoptosis,ubiquitin cycle,		apoptosis,ubiquitination,inhibition of apoptosis,drug resistance,
nr1_gene				oxidation-reduction,
interleukin_gene				cytokine signaling,intercellular communication,cell proliferation regulation,
pdap2_gene			3p21.3	
transcription_factor_gene				transcriptional regulation,
bai2_gene	1p35	neuropeptide signaling pathway,signal transduction,	1p35	angiogenesis,
transcription_coregulator_gene				transcriptional regulation,
bcl2l11_gene	2q13	apoptosis,induction of apoptosis,positive regulation of apoptosis,		apoptosis,induction of apoptosis,

NCBI Gene Name	NCBI Chromosome Location	NCBI Process list	NCI Chromosome Location	NCI Process list
mmp2_gene	16q13-q21	collagen catabolism,peptidoglycan metabolism,proteolysis,		inflammation process,proteolysis,angiogenesis,
rfp2_gene	13q14	cell cycle,morphogenesis,negative regulation of progression through cell cycle,positive regulation of i-kappab kinase/nf-kappab cascade,	13q14	
gprk2l_gene			4p16.3	signal transduction,phosphorylation,g protein-coupled receptor signaling,serine-threonine phosphorylation,
il1a_gene	2q14	anti-apoptosis,apoptosis,cell proliferation,cell-cell signaling,chemotaxis,fever,negative regulation of cell proliferation,regulation of progression through cell cycle,	2q14	immune response,cell proliferation regulation,induction of apoptosis,hematopoiesis,intercellular communication,cytokine signaling,inflammatory response,
smarca1_gene	xq25	chromatin remodeling,regulation of transcription,dna-dependent,transcription,	xq25	chromatin remodeling,transcriptional regulation,

NCBI Gene Name	NCBI Chromosome Location	NCBI Process list	NCI Chromosome Location	NCI Process list
hmgb1_gene	13q12	dna recombination,dna repair,dna unwinding during replication,anti-apoptosis,base-excision repair,dna ligation,establishment and/or maintenance of chromatin architecture,negative regulation of transcriptional preinitiation complex formation,regulation of transcription from rna polymerase ii promoter,regulation of transcription,dna-dependent,signal transduction,	13q12	v-d-j recombination,dna binding,protein-protein interaction,transcription,dna repair,transcription initiation,
ceacam5_gene	19q13.1-q13.2		19q13.1-q13.2	

NCBI Gene Name	NCBI Chromosome Location	NCBI Process list	NCI Chromosome Location	NCI Process list
ppbp_gene	4q12-q13	cell proliferation,chemotaxis,defense response to bacteria,glucose transport,immune response,regulation of progression through cell cycle,sensory perception,		immune response,cell-matrix adhesion,intercellular communication,signal transduction,glycolysis,immunoregulation,chemotaxis,cytokine signaling,mitosis,
msx1_gene	4p16.3-p16.1	development,organ morphogenesis,regulation of transcription,dna-dependent,skeletal development,	4p16.3-p16.1	transcriptional repression,cell differentiation,pattern formation,

NCBI Gene Name	NCBI Chromosome Location	NCBI Process list	NCI Chromosome Location	NCI Process list
tap2_gene	6p21.3	antigen presentation,endogenous peptide antigen,antigen presentation,endogenous peptide antigen,antigen processing,endogenous antigen via mhc class i,antigen processing,endogenous antigen via mhc class i,cytosol to er transport,cytosol to er transport,immune response,intracellular protein transport,intracellular protein transport,oligopeptide transport,peptide transport,protein complex assembly,protein complex assembly,	6p21.3	protein-protein interaction,multidrug resistance,immune surveillance,antigen presentation,drug efflux,ligand binding,peptide transport,
csf3r_gene	1p35-p34.3	cell adhesion,defense response,signal transduction,	1p35-p34.3	ligand binding,intercellular communication,receptor signaling,cell adhesion,cell differentiation,

NCBI Gene Name	NCBI Chromosome Location	NCBI Process list	NCI Chromosome Location	NCI Process list
mpp1_gene	xq28	signal transduction,	xq28	phosphorylation,signal transduction,cytoskeletal modeling,intercellular communication,cell proliferation,
en1_gene	11q13	negative regulation of transcription from rna polymerase ii promoter,regulation of transcription,dna-dependent,	11q13	tumor suppression,transcriptional repression,

APPENDIX D

ALGORITHM

For every gene with n processes in the NCIT's Biological Process (BP) hierarchy:

If c1 is a process for gene g in NCBI,

and c2 is a process for g in NCIT.

Case 1 : No Action, same concepts found in both databases

If c1 is a concept in BP equal to c2, take no action.

Case 2 : Replace existing concept with child concept

If c1 is a concept in BP and c2 is a parent of c1

Then replace c2 with c1, since c1 is a more

refined target for g in NCIT.

Case 3 : Replace concept with synonym of found concept

If c1 is a synonym of concept c3 in BP and c2 is a

parent of c3

Then replace c2 with c3 for g in NCIT since c1 is a

more refined target for g in NCIT.

Case 4 :Addition of existing concept

If c1 is a concept or a synonym in BP for g,

then add c1 to the processes of g in NCIT

Case 5 : Addition of new concept to BP and gene role target

If no previous condition is satisified,

then add c1 to BP and to gene role target for g.

APPENDIX E

JAVA PROGRAM TO COLLECT DATA

```
package ontology;

import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.FileNameMap;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;
import java.net.UnknownHostException;
import java.util.Enumeration;
import java.util.Hashtable;
import java.util.Properties;
import java.util.StringTokenizer;
import java.util.Vector;

/*
 *
 * @author marc
 *
 * Formats data from NCBI and NCI databases into 5 columns:
```

```
 *  gene name | associated with disease | found in organism | In chromosomal location | Plays Role in Process
 */
public class DownloadHTMLFile {

        // NCBI variables
        static URL NCBIUrl = null;

        static URL secondaryNCBIUrl = null;

        static String NCBIbaseURL = null;

        static String[] WebInfo = null;

        static String NCBIqueryString = null;

        static Properties prop = null;

        static Vector bycode = null;

        static Vector byname = null;

        static String current = null; // current name

        static String currentCode = null; // current code

        static Hashtable htbl = null;

        // NCI variables
        static URL NCIUrl = null;

        static URL NCIStaticUrl = null;
```

```
static String NCIStaticUrlString = null;

static URL secondaryNCIUrl = null;

static String NCIbaseURL = null;
static String NCI2baseURL = null;

static String NCIqueryString = null;

static String NCIqueryString2 = null;

static String NCIjsessionId = null;

static logwriter output = null;

static logwriter systemOut = null;

static String ncbiCurrent = null;

static URLConnection NCBIconnection = null;

static URLConnection NCIconnection = null;

static HttpURLConnection NCI2connection = null;
static HttpURLConnection h = null;

static String login_cookie = "";


public lineRecord ncbi = null;
```

```
public lineRecord nci = null;

// set to true to write debugging data
static boolean debug = false;
static boolean displayParent = false;
static boolean printNCBI      = true;
static boolean printNCI                = true;

static int    childCnt    = 0;
static int    parentCnt   = 0;
static int    synCnt    = 0;


public static void main(String[] args) {
        try {
                if (!getInputParameters(args)) {
                        return;
                }
                setNCBIbaseURL(prop.getProperty("NCBIBaseURL"));
                setNCBIqueryString(prop.getProperty("NCBIQuery"));
                setNCIbaseURL(prop.getProperty("NCIBaseURL"));
                setNCI2baseURL(prop.getProperty("NCI2BaseURL"));
                setNCIqueryString(prop.getProperty("NCIQuery"));
                setNCIqueryString2(prop.getProperty("NCIQuery2"));
                setNCIStaticUrlString(prop.getProperty("NCIConnect"));

                if (prop.getProperty("ProxyHost") != null
                                && prop.getProperty("ProxyPort") != null) {
                        System.setProperty("proxyHost", prop.getProperty("ProxyHost"));
                        System.setProperty("proxyPort", prop.getProperty("ProxyPort"));
```

```
}

if (prop.getProperty("debug") != null
                && prop.getProperty("debug").equals("Y")) {
        debug = true;
        systemOut = new logwriter();
        systemOut.init(prop.getProperty("outputDir"), "DebugFile");
}

if (prop.getProperty("displayParent") != null && prop.getProperty("displayParent").equals("Y")) {
        displayParent= true;
}

if (prop.getProperty("printNCBI") != null && prop.getProperty("printNCBI").equals("N")) {
        printNCBI= false;
}
if (prop.getProperty("printNCI") != null && prop.getProperty("printNCI").equals("N")) {
        printNCI= false;
}

htbl = new Hashtable();
setupWriter();

Enumeration e = byname.elements();
Enumeration eCode = bycode.elements();
while (e.hasMoreElements() && eCode.hasMoreElements()){

current = (String) e.nextElement();
if (current.indexOf("_Gene") > 0) {
        ncbiCurrent = current.substring(0, current.indexOf("_Gene"));
        ncbiCurrent = ncbiCurrent.replace("_Gene", "");
```

```
                } else
                        ncbiCurrent = current;

                getNCBIConn(getNCBIbaseURL() + getNCBIqueryString() + ncbiCurrent);
                getNCBIData();

                currentCode = (String) eCode.nextElement();
                // getNCIConn(getNCIbaseURL() + getNCIqueryString() + currentCode );
                // getNCIData(currentCode);
//              getNCIDataFromPost("C18120");
                getNCIDataFromPost(currentCode);
                // testPost();
                printOneRecord(output, current);
                } // end while
                // printRecords(output);
        } catch (Exception e) {
                e.printStackTrace();
                output.closeLogFile();
        }
}

public static boolean getInputParameters(String[] args) {
        try {
                // Open the file that is the first
                // command line parameter
                if (args[0].equalsIgnoreCase("-h")) {
                        System.out.println("-p PathToPropertyFile");
                        return false;
                }

                if (args[0].equalsIgnoreCase("-p")) {
```

```
        prop = new Properties();
        try {
                FileInputStream fis = new FileInputStream(args[1]);
                prop.load(fis);
                System.out.println(prop.toString());
        } catch (Exception e) {
                System.err.println("Could not open properties file -p filename");
                return false;
        }
}

FileInputStream fstream = new FileInputStream(prop
                .getProperty("GenesFile"));
DataInputStream in = new DataInputStream(fstream);
byname = new Vector();
bycode = new Vector();
String tmp = "";
while (in.available() != 0) {
        // Print file line to screen
        // System.out.println(in.readLine());
        StringTokenizer st = new StringTokenizer(in.readLine());
        bycode.addElement(st.nextToken());
        byname.addElement(st.nextToken());
}
in.close();

// Convert our input stream to a
// DataInputStream

// Continue to read lines while
// there are still some left to read
```

```
        } catch (Exception e) {
                System.err.println("File input error");
                return false;
        }

        return true;
}

public static void setupWriter() {
        output = new logwriter();
        output.init(prop.getProperty("outputDir"), prop
                        .getProperty("outputFileName"));
}

public static void printRecords(logwriter o) {
        Enumeration e = htbl.elements();
        lineRecord l = new lineRecord();

        while (e.hasMoreElements()) {
                l = (lineRecord) e.nextElement();
                StringBuffer sb = new StringBuffer();

                int start = l.getGeneName().indexOf(".");
                sb.append(l.getGeneName().substring(start + 1));
                sb.append(" ;");
                for (int i = 0; i < l.getChromLocSize()
                                && l.getChromosomeLocation(i) != null; i++) {
                        sb.append(l.getChromosomeLocation(i));
                        sb.append(" ");
                }
```

```
                sb.append(" ;");
                for (int i = 0; i < l.getProcessSize() && l.getProcesses(i) != null; i++) {
                        sb.append(l.getProcesses(i));
                        sb.append(", ");
                }
                sb.append(" ;");
                o.log(sb);
        }

}

public static void printOneRecord(logwriter o, String geneName) {

        try {
        lineRecord l = (lineRecord) htbl.get("ncbi." + geneName);
                StringBuffer sb = new StringBuffer();

                // leave spaces to allow for string tokenizer to work properly in datacompare.java
                if (printNCBI){
                        if (l != null) {

                                int start = l.getGeneName().indexOf(".");
                                sb.append(l.getGeneName().substring(start + 1));
                                sb.append("; ");
                                for (int i = 0; i < l.getChromLocSize()
                                                && l.getChromosomeLocation(i) != null; i++) {
                                        sb.append(l.getChromosomeLocation(i));
                                        sb.append(" ");
                                }
                                sb.append("; ");
                                for (int i = 0; i < l.getProcessSize()
```

```
                                && l.getProcesses(i) != null; i++) {
                        sb.append(l.getProcesses(i));
                        sb.append(", ");
                }
                sb.append("; ");
                for (int i = 0; i < l.getDiseaseSize()
                                && l.getDiseases(i) != null; i++) {
                        sb.append(l.getDiseases(i));
                        sb.append(", ");
                }
                sb.append("; ");
        } else {
                sb.append(geneName + "; ; ; ; ");
        }
}

l = (lineRecord) htbl.get("nci." + geneName);

if (printNCI) {
        if (l != null) {

                // check if we found a genename before moving the index.
                if (l.getGeneName().length() > 0 && l.getGeneName() != null){
                        int start = l.getGeneName().indexOf(".");
                        sb.append(l.getGeneName().substring(start + 1));
//                      sb.append(l.getGeneName());
                }
                sb.append("; ");
                for (int i = 0; i < l.getChromLocSize()
                                && l.getChromosomeLocation(i) != null; i++) {
                        if (l.getChromosomeLocation(i).substring(0,1).equals(".")) {
```

```
sb.append(l.getChromosomeLocation(i).substring(1,l.getChromosomeLocation(i).length()));
                        sb.append(" ");
                }
                else{
                        sb.append(l.getChromosomeLocation(i));
                        sb.append(" ");
                }
        }
        sb.append("; ");
        for (int i = 0; i < l.getProcessSize()
                        && l.getProcesses(i) != null; i++) {
                sb.append(l.getProcesses(i));
                sb.append(", ");
        }
        sb.append("; ");
        for (int i = 0; i < l.getDiseaseSize()
                        && l.getDiseases(i) != null; i++) {
                sb.append(l.getDiseases(i));
                sb.append(", ");
        }
        sb.append("; ");
        for (int i = 0; i < l.getSuperConceptSize()
                        && l.getSuperconcepts(i) != null; i++) {
                sb.append(l.getSuperconcepts(i));
                sb.append(", ");
        }
        sb.append("; ");
//      for (int j = 0; j < l.getProcessSize()&& l.getProcesses(j) != null; j++) {
                for (int i = 0; i < l.getSubConceptSize() && l.getSubconcepts(i) != null; i++) {
                        sb.append(l.getSubconcepts(i));
```

```
                                        sb.append(", ");
                                }
//                              }
                                sb.append("; ");
                                for (int i = 0; i < l.getSynConceptSize()
                                        && l.getSynconcepts(i) != null; i++) {
                                sb.append(l.getSynconcepts(i));
                                sb.append(", ");
                                }
                                sb.append("; ");


                        } else {
                                sb.append(geneName + "; ; ; ; ");
                        }
                }
                o.log(sb);

        } catch (Exception e) {
                System.out.println("There was an error printing Gene: " + geneName);
                e.printStackTrace();
        }

}

public static void getNCBIConn(String website) {
        try {
                NCBIUrl = new URL(website);
        } catch (MalformedURLException e) {
                // Malformed URL
                System.out.println("Error in given URL");
```

```
                return;
        }
}

public static URLConnection getNCIConn(String website, URLConnection nciConn) {
        try {

                NCIUrl = new URL(website);
                // NCIStaticUrl = new URL(getNCIStaticUrlString());
                return NCIUrl.openConnection();
                // NCIconnection.setAllowUserInteraction(true);

                // NCIjsessionId = NCIconnection.getRequestProperty("jsessionid");
                /*
                 * BufferedReader br = new BufferedReader(new
                 * InputStreamReader(NCIconnection.getInputStream())); String line =
                 * ""; while ( ( line = br.readLine()) != null){ if
                 * (line.contains("jsessionid=")){ int start=
                 * line.indexOf("jsessionid=") + 11; if (start > 0){ int end =
                 * line.indexOf("\" ",start); if (! (end > 0)){ NCIjsessionId =
                 * line.substring(start); break; } else{ NCIjsessionId =
                 * line.substring(start,end); break; } } } //
                 * System.out.println(line); } String tmp =
                 * website+"&jsessionid="+NCIjsessionId; NCIUrl = new URL(tmp);
                 */} catch (MalformedURLException e) {
                // Malformed URL
                System.out.println("Error in given URL");
                return null;
        } catch (IOException ioe) {
                // IOException URL
                System.out.println("IO exception in static URL");
```

```
                return null;
        }
}

public static void getNCBIData() {
        try {
                NCBIconnection = NCBIUrl.openConnection();
                BufferedReader br = new BufferedReader(new InputStreamReader(
                                NCBIconnection.getInputStream()));
                String line = "";
                String link = "";
                // assign line to lastline for finding links.
                while ((line = br.readLine()) != null) {
                        // System.out.println(line);
                        if (line.contains("a href=\"/entrez")) {
                                int linkStart = line.indexOf("href=") + 6;
                                if (linkStart > 0) {
                                        int linkEnd = line.indexOf("\">", linkStart);
                                        if (!(linkEnd > 0))
                                                link = line.substring(linkStart);
                                        else
                                                link = line.substring(linkStart, linkEnd);
                                }
                        }

                        if ((line.contains("Official Symbol:"))
                                        && (line.contains("Homo sapiens"))) {
                                int symbolStart = line.indexOf("</b> ") + 5;
                                int symbolEnd = line.indexOf(" <b>", symbolStart);
                                String pageSymbol = line.substring(symbolStart, symbolEnd);
                                // System.out.println(line);
```

```
if (ncbiCurrent.equalsIgnoreCase(pageSymbol)) {

        lineRecord ncbi = new lineRecord(Integer.parseInt(prop
                .getProperty("defaultNumberOfWebFields")));
        ncbi.setGeneName("ncbi." + current);
        ncbi.setOrganism("Homo Sapiens", 0);
        htbl.put("ncbi." + current, ncbi);
        for (int j = 0; j < 4; j++) {
            line = br.readLine();
            if (line.contains("Location:")) {
                int chromosomeStart = line.indexOf("</b>") + 4;
                int chromosomeEnd = line.indexOf(";");
                // System.out.println("chromosome = " +
                // line.substring(chromosomeStart,chromosomeEnd));
                // log("chromosome",line.substring(chromosomeStart,chromosomeEnd));
                int locationStart = line.indexOf("</b>",
                        chromosomeEnd) + 4;
                int locationEnd = line.indexOf("</dd>",
                        locationStart);
                // log( "location",
                // line.substring(locationStart,locationEnd));
                //
ncbi.setChromosomeLocation(line.substring(chromosomeStart,chromosomeEnd)
                // + " " +
                // line.substring(locationStart,locationEnd),
                // 0);
                ncbi.setChromosomeLocation(line.substring(
                        locationStart, locationEnd), 0);
                // System.out.println("link line = " + link);
                // log("link",link);
                if (get2ndNCBIConn(NCBIbaseURL + link)) {
```

```
                                                    get2ndNCBIData(ncbi);
                                                }
                                            } // end if

                                        } // end for

                                    }
                                }

                            }

                            br.close();
                        } catch (UnknownHostException e) {
                            System.out.println("Unknown Host");
                            return;
                        } catch (IOException e) {
                            System.out.println("Error in opening URLConnection, Reading or Writing for NCBI.  URL = " +
NCBIUrl.toString());
                            return;
                        }
                }

                /*
                 * pass log a key value pair to write it to the file
                 */
                public static void log(String key, String value) {
                        System.out.println(key + ":" + value);
                }

                public static boolean get2ndNCBIConn(String website) {
                        try {
```

```
                secondaryNCBIUrl = new URL(website);
                return true;
        } catch (MalformedURLException e) {
                // Malformed URL
                System.out.println("Error in given URL" + secondaryNCBIUrl);
                return false;
        }
}

/**
 * @return Returns the NCBIbaseURL.
 */
public static String getNCBIbaseURL() {
        return NCBIbaseURL;
}

/**
 * @param NCBIbaseURL
 *          The NCBIbaseURL to set.
 */
public static void setNCBIbaseURL(String baseURL) {
        DownloadHTMLFile.NCBIbaseURL = baseURL;
}

/**
 * @return Returns the NCBIqueryString.
 */
public static String getNCBIqueryString() {
        return NCBIqueryString;
}
```

```
/**
 * @param NCBIqueryString
 *            The NCBIqueryString to set.
 */
public static void setNCBIqueryString(String queryString) {
        DownloadHTMLFile.NCBIqueryString = queryString;
}

public static boolean get2ndNCBIData(lineRecord ncbi) {
        try {
                URLConnection connection = secondaryNCBIUrl.openConnection();
                BufferedReader br = new BufferedReader(new InputStreamReader(
                                connection.getInputStream()));
                String line = "";
                String link = "";
                // assign line to lastline for finding links.
                while ((line = br.readLine()) != null) {
                        // System.out.println(line);
                        if (line.contains("a href=\"/entrez")) {
                                int linkStart = line.indexOf("href=") + 6;
                                if (linkStart > 0) {
                                        int linkEnd = line.indexOf("\">", linkStart);
                                        if (!(linkEnd > 0))
                                                link = line.substring(linkStart);
                                        else
                                                link = line.substring(linkStart, linkEnd);
                                }
                        }

                        if ((line.contains("<b>Process</b>"))) {
                                // System.out.println(line);
```

```
                        int i = 0;
                        while (!(line.contains("</table>"))) {
                                if ((line.contains("Component"))
                                                || (line.contains("Homology"))) {
                                        break;
                                }
                                line = br.readLine();
                                if (line.contains("\">")) {
                                        int processStart = line.indexOf("\">") + 2;
                                        int processEnd = line.indexOf("</a>", processStart);
                                        // System.out.println("process = " +
                                        // line.substring(processStart,processEnd));
                                        ncbi.setProcesses(line.substring(processStart,
                                                        processEnd), i);
                                        i++;
                                }
                        }
                }

        }
        br.close();
        return true;
} catch (UnknownHostException e) {
        System.out.println("Unknown Host");
        return false;
} catch (IOException e) {
        System.out
                        .println("Error in opening URLConnection, Reading or Writing for 2nd page of NCBI data.
URL = " + secondaryNCBIUrl);
        return false;
}
```

```
}

public static void getNCIData(String current) {
        try {
                NCIconnection = getNCIConn(getNCIbaseURL() + getNCIqueryString() + current
                                + getNCIqueryString2(), NCIconnection);

                // NCI2connection = NCIUrl.openConnection();

                BufferedReader br = new BufferedReader(new InputStreamReader(
                                NCIconnection.getInputStream()));
                String line = "";
                String tmp = "";
                int geneCnt = 0;
                int processCnt = 0;
                int diseaseCnt = 0;

                lineRecord lr = new lineRecord(Integer.parseInt(prop
                                .getProperty("defaultNumberOfWebFields")));
                while ((line = br.readLine()) != null) {
                        if (debug) {
                                systemOut.log(line.toString());
                        }
                        /*
                         * if (line.contains("name")){ for (int c = 0; (line =
                         * br.readLine()) != null; c++ ) { if (c == 3) { tmp =
                         * line.trim(); int start= line.indexOf("\">") + 2; if (start >
                         * 0){ int end = line.indexOf("</",start); if (! (end > 0)){
                         * tmp = line.substring(start); } else{ tmp =
                         * line.substring(start,end); } } lr.setGeneName("nci."+ tmp);
                         * htbl.put(lr.getGeneName(),lr); break; } // end-if } // end
```

```
                    * for } // end-if
                    *
                    * if (line.contains("Gene_In_Chromosomal_Location")){ for (int
                    * c = 0; (line = br.readLine()) != null; c++ ) { if (c == 5) {
                    * tmp = line.trim(); lr.setChromosomeLocation(tmp,geneCnt);
                    * geneCnt++; break; } // end-if } // end for } // end-if
                    *
                    * if (line.contains("Gene_Associated_With_Disease")){ for (int
                    * c = 0; (line = br.readLine()) != null; c++ ) { if (c == 5) {
                    * tmp = line.trim(); lr.setDiseases(tmp,diseaseCnt);
                    * diseaseCnt++; break; } // end-if } // end for } // end-if
                    *
                    * if (line.contains("Gene_Plays_Role_in_Process")){ for (int c =
                    * 0; (line = br.readLine()) != null; c++ ) { if (c == 5) { tmp =
                    * line.trim(); lr.setProcesses(tmp,processCnt); processCnt++;
                    * break; } // end-if } // end for } // end-if
                    */
            } // end while

            br.close();
    } catch (UnknownHostException e) {
            System.out.println("Unknown Host");
            /*
             * int respCode = ((HttpURLConnection)conn).getResponseCode(); es =
             * ((HttpURLConnection)conn).getErrorStream(); int ret = 0; // read
             * the response body while ((ret = es.read(buf)) > 0) {
             * processBuf(buf); } // close the errorstream es.close();
             */
            return;
    } catch (IOException e) {
            e.printStackTrace();
```

```
                System.out
                                .println("Error in opening URLConnection, Reading or Writing");
                return;
        }
}

public static void getNCIDataFromPost(String current) {
        try {
                NCIconnection = getNCIConn(getNCIbaseURL() + getNCIqueryString() + current
                                + getNCIqueryString2(), NCIconnection);

                // NCI2connection = NCIUrl.openConnection();

                BufferedReader br = new BufferedReader(new InputStreamReader(
                                NCIconnection.getInputStream()));
                String line = "";
                String tmp = "";
                int geneCnt = 0;
                int processCnt = 0;
                int diseaseCnt = 0;

                childCnt = 0;
                parentCnt = 0;
                synCnt = 0;

                lineRecord lr = new lineRecord(Integer.parseInt(prop
                                .getProperty("defaultNumberOfWebFields")));
                while ((line = br.readLine()) != null) {
                        if (debug) {
                        //      systemOut.log(line.toString());
                        }
```

```
// if (line.contains("name")){
if (line.contains("Identifiers:")) {
        for (int c = 0; (line = br.readLine()) != null; c++) {
                if (c == 8) {
                        tmp = line.trim();
                        /*
                         * int start= line.indexOf("\">") + 2; if (start >
                         * 0){ int end = line.indexOf("</",start); if (!
                         * (end > 0)){ tmp = line.substring(start); } else{
                         * tmp = line.substring(start,end); } }
                         */
                        lr.setGeneName("nci." + tmp);
                        htbl.put(lr.getGeneName(), lr);
                        break;
                } // end-if
        } // end for
} // end-if

if (line.contains("Gene_In_Chromosomal_Location") && line.contains("group_number") != true) {
        for (int c = 0; (line = br.readLine()) != null; c++) {
                if (getGenes( br, lr, line)){
                        break;
                }
        } // end for
} // end-if

if (line.contains("Gene_Associated_With_Disease") && line.contains("group_number") != true) {
        for (int c = 0; (line = br.readLine()) != null; c++) {
                if (getDiseases(diseaseCnt, br, lr, line) > diseaseCnt){
                        diseaseCnt++;
```

```
                                        break;
                                    }
                                } // end for
                            } // end-if

                            if (line.contains("Gene_Plays_Role_in_Process") && line.contains("group_number") != true) {
                                for (int c = 0; (line = br.readLine()) != null; c++) {
                                    if (getProcesses(processCnt, br, lr, line) > processCnt){
                                        processCnt++;
                                        break;
                                    }
                                }//end for
                            } // end-if

                        } // end while

                        br.close();
                    } catch (UnknownHostException e) {
                        System.out.println("Unknown Host");
                        /*
                         * int respCode = ((HttpURLConnection)conn).getResponseCode(); es =
                         * ((HttpURLConnection)conn).getErrorStream(); int ret = 0; // read
                         * the response body while ((ret = es.read(buf)) > 0) {
                         * processBuf(buf); } // close the errorstream es.close();
                         */
                        return;
                    } catch (IOException e) {
                        e.printStackTrace();
                        System.out.println("Error in opening URLConnection, Reading or Writing URL = " + getNCIbaseURL() +
getNCIqueryString() + current + getNCIqueryString2());
                        return;
```

```
        }
        catch (Exception ez)
        {
                ez.printStackTrace();
        }
}

/**
 * @return Returns the output.
 */
public logwriter getOutput() {
        return output;
}

/**
 * @param output
 *          The output to set.
 */
public void setOutput(logwriter output) {
        this.output = output;
}

/**
 * @return Returns the nCIbaseURL.
 */
public static String getNCIbaseURL() {
        return NCIbaseURL;
}

/**
 * @param ibaseURL
```

```
     *          The nCIbaseURL to set.
     */
    public static void setNCIbaseURL(String ibaseURL) {
            NCIbaseURL = ibaseURL;
    }

    /**
     * @return Returns the nCIqueryString.
     */
    public static String getNCIqueryString() {
            return NCIqueryString;
    }

    /**
     * @param iqueryString
     *          The nCIqueryString to set.
     */
    public static void setNCIqueryString(String iqueryString) {
            NCIqueryString = iqueryString;
    }

    /**
     * @return Returns the nCIStaticUrlString.
     */
    public static String getNCIStaticUrlString() {
            return NCIStaticUrlString;
    }

    /**
     * @param staticUrlString
     *          The nCIStaticUrlString to set.
```

```
*/
public static void setNCIStaticUrlString(String staticUrlString) {
        NCIStaticUrlString = staticUrlString;
}

public static String getNCIqueryString2() {
        return NCIqueryString2;
}

public static void setNCIqueryString2(String iqueryString2) {
        NCIqueryString2 = iqueryString2;
}

public static void getIdentifier(String iqueryString2, BufferedReader br,lineRecord lr, String line) {

        String tmp = "";
        // if (line.contains("name")){
        try {
                if (line.contains("Identifiers:")) {
                        for (int c = 0; (line = br.readLine()) != null; c++) {
                                if (c == 8) {
                                        tmp = line.trim();
                                        /*
                                         * int start= line.indexOf("\">") + 2; if (start > 0){ int
                                         * end = line.indexOf("</",start); if (! (end > 0)){ tmp =
                                         * line.substring(start); } else{ tmp =
                                         * line.substring(start,end); } }
                                         */
                                        lr.setGeneName("nci." + tmp);
                                        htbl.put(lr.getGeneName(), lr);
                                        break;
```

```
                        } // end-if
                    } // end for
                } // end-if
        }
        catch (IOException ioe){
                ioe.printStackTrace();
        }
} // end getIdentifier

public static int getProcesses(int processCnt, BufferedReader br, lineRecord lr,String line) {

        String tmp = "";
        String tmpName = "";
        // if (line.contains("name")){

        if (line.contains("GetRoleAndProperty.do")) {
                String href = getHref(line);
                int start = line.lastIndexOf("conceptname=");
                if (start > 0) {
                        int end = line.indexOf("\"", start);
                        if (!(end > 0)) {
                                tmp = line.substring(start + "conceptname=".length());
                        } else {
                                tmp = line.substring(start + "conceptname=".length(), end);
                        }
                        // only set if conceptname found in page
                        tmpName = tmpReplace(tmp);
                        lr.setProcesses(tmpName, processCnt);
                        if (href != null && href.length() > 0 ){

                                getParentChildProcesses(tmp,lr,processCnt);
```

```
                                        //                                  getParentChildProcesses(currentCode,lr);
//                                      checkProcessVsNCBI(tmpName,lr,processCnt);
//                                      addProcessesFromNCBI(tmpName,lr,processCnt);
                                }
                                processCnt++;
                        }
                } // end-if
                return processCnt;
        } // end getProcesses

        public static boolean getGenes(BufferedReader br, lineRecord lr,String line) {

                String tmp = "";
                // if (line.contains("name")){
                if (line.contains("GetRoleAndProperty.do")) {
                        int start = line.lastIndexOf("conceptname=");
                        if (start > 0) {
                                int end = line.indexOf("\"", start);
                                if (!(end > 0)) {
                                        tmp = line.substring(start + "conceptname=".length());
                                } else {
                                        tmp = line.substring(start + "conceptname=".length(), end);
                                }
                                // only set if conceptname found in page
                                if (tmp.substring(0).equals("_")){
                                        tmp = tmp.replaceFirst("_","");
                                }
                                lr.setChromosomeLocation(tmp.replace("_",".."),0);
                        }
                        return true;
                } // end-if
```

```
        else
                return false;
} // end getIdentifier


public static int getDiseases(int  diseaseCnt, BufferedReader br, lineRecord lr ,String line) {

        String tmp = "";

        if (line.contains("GetRoleAndProperty.do")) {
                int start = line.lastIndexOf("conceptname=");
                if (start > 0) {
                        int end = line.indexOf("\"", start);
                        if (!(end > 0)) {
                                tmp = line.substring(start + "conceptname=".length());
                        } else {
                                tmp = line.substring(start + "conceptname=".length(), end);
                        }
                        // only set if conceptname found in page
                        tmp = tmpReplace(tmp);
                        lr.setDiseases(tmp, diseaseCnt);
                        diseaseCnt++;
                }
        } // end-if
        return diseaseCnt;
} // end getDiseases

public static String getHref(String line) {

        String tmp = "";
```

```
                if (line.contains("GetRoleAndProperty.do")) {
                        int start = line.indexOf("GetRoleAndProperty.do");
                        if (start > 0) {
                                int end = line.indexOf("\"", start);
                                if (!(end > 0)) {
                                        tmp = line.substring(start);
                                } else {
                                        tmp = line.substring(start, end);
                                }
                        }
                } // end-if
                return tmp;
        } // end getHref

        public static  void getParentChildProcesses(String href, lineRecord lr, int processCnt){
                try {
/*                      if (NCI2connection == null){
                                NCI2connection = (HttpURLConnection)getNCIConn(prop.getProperty("NCIConnect"),
NCI2connection);
                                NCI2connection.setDoOutput(true);
                                NCI2connection.setDoInput(true);
                                NCI2connection.setFollowRedirects(true);
//                              NCI2connection.setRequestMethod("POST");
                                NCI2connection.setAllowUserInteraction(false);
                        } // end if

                        if (! NCI2connection.equals(null)){
                                login_cookie = NCI2connection.getHeaderField("Set-Cookie");
                                if (login_cookie == null) {
                                        System.out.println ("COOKIE NOT FOUND");
                                        return;
```

```
                } else {System.out.println ("login_cookie: " + login_cookie);}
            }
            else{
                return;
            }

            int start = login_cookie.indexOf("JSESSIONID=") + "JSESSIONID=".length();
            int end   = login_cookie.indexOf(";");
            String jsessionid = login_cookie.substring(start,end);
*/
            try{
                h = (HttpURLConnection)getNCIConn(getNCI2baseURL() + href, h);
            }
            catch (Exception i){

            }

            h.setDoOutput(true);
            h.setRequestMethod("GET");

            h.setAllowUserInteraction(false);
            h.setDoInput(true);
/*          h.setRequestProperty("bookmarktag","1");

        start = href.indexOf("conceptname=") + "conceptname=".length();
            end   = href.indexOf("\"");

            h.setRequestProperty("pattern1",href.substring(start));
            if (login_cookie != null){
                h.setRequestProperty("Cookie", login_cookie);
            }
```

```
h.setRequestProperty( "Content-length", Integer.toString(h.getContentLength()));

PrintWriter out = new PrintWriter(h.getOutputStream());
out.println(h.getContent());
out.flush();
out.close();

*/

BufferedReader br = new BufferedReader(new InputStreamReader(h.getInputStream()));
String line = "";
String tmp = "";

boolean superFlag    = false;
boolean subFlag      = false;
//syn = synonyms
boolean synFlag      = false;

int i = 0;
while ((line = br.readLine()) != null) {
        if (debug) {
                systemOut.log(line.toString());
        }
        if (line.contains("Identifiers")){
                System.out.println("Identifiers");
        }
//      System.out.println(line.toString());
        // if (line.contains("name")){
        if (line.contains("Superconcepts")) {
                superFlag=true;
                subFlag = false;
                synFlag = false;
```

```
}
if (line.contains("Subconcepts")) {
        superFlag=false;
        subFlag = true;
        synFlag = false;
}
if (line.contains("Information about this concept:")) {
        superFlag=false;
        subFlag = false;
        synFlag = true;
}

if (superFlag){
        parentCnt = getSuperConcepts(parentCnt,br,lr,line, processCnt);
}
else if (subFlag){
        childCnt = getSubConcepts(childCnt,br,lr,line, processCnt);
}
else if (synFlag){
        synCnt = getSynConcepts(synCnt,br,lr,line, processCnt);
}

//      break out of loop
/*      if (line.contains("application footer begins")){
        break;
}

*/
if ( (superFlag || subFlag) && line.contains("</table>")){
        break;
} // end if
i++;
```

```
                        } // end while

                        System.out.println("# of Lines = " + i);
                        br.close();
                } catch (UnknownHostException e) {
                        System.out.println("Unknown Host");
                        /*
                         * int respCode = ((HttpURLConnection)conn).getResponseCode(); es =
                         * ((HttpURLConnection)conn).getErrorStream(); int ret = 0; // read
                         * the response body while ((ret = es.read(buf)) > 0) {
                         * processBuf(buf); } // close the errorstream es.close();
                         */
                        return;
                } catch (IOException e) {
                        e.printStackTrace();
                        System.out.println("Error in opening URLConnection, Reading or Writing URL = " + getNCIbaseURL() +
getNCIqueryString() + current + getNCIqueryString2());
                        return;
                }
                catch (Exception ez)
                {
                        ez.printStackTrace();
                }

        } // end getParentChildProcesses

        public static int getSubConcepts(int  childCnt, BufferedReader br, lineRecord lr ,String line, int processCnt) {

                String tmp = "";

                // check that there are no tags and the line has some text on it
```

```
            if ( (!line.contains("<") && !line.contains(">") ) && !line.trim().equals("")){
                    tmp = line.trim();
                    lr.setSubconcepts(processCnt + "."  + tmp,childCnt);
                    childCnt++;
            } // end if

/*          if (line.contains("GetRoleAndProperty.do")) {
                    int start = line.lastIndexOf("conceptname=");
                    if (start > 0) {
                            int end = line.indexOf("\"", start);
                            if (!(end > 0)) {
                                    tmp = line.substring(start + "conceptname=".length());
                            } else {
                                    tmp = line.substring(start + "conceptname=".length(), end);
                            }
                            // only set if conceptname found in page
                            tmp = tmpReplace(tmp);
                            lr.setSubconcepts(tmp.replaceAll("_"," "), childCnt);
                            childCnt++;
                    }
            } // end-if
*/
            return childCnt;
    } // end getSubConcepts

    public static int getSynConcepts(int  synCnt, BufferedReader br, lineRecord lr ,String line, int processCnt) {

            String tmp = "";

            // check that there are no tags and the line has some text on it
            if ( (!line.contains("<") && !line.contains(">") && !line.contains("|") && !line.contains("Synonym") &&
```

```
!line.contains("Unified") && !line.contains("NCI_") && !line.contains("DEFINITION") && !line.contains("Semantic_Type") &&
!line.contains("Preferred_Name")) && !line.trim().equals("")){
                        tmp = line.trim();
                        lr.setSynconcepts(processCnt + "."  + tmp,synCnt);
                        synCnt++;
                } // end if

/*              if (line.contains("GetRoleAndProperty.do")) {
                        int start = line.lastIndexOf("conceptname=");
                        if (start > 0) {
                                int end = line.indexOf("\"", start);
                                if (!(end > 0)) {
                                        tmp = line.substring(start + "conceptname=".length());
                                } else {
                                        tmp = line.substring(start + "conceptname=".length(), end);
                                }
                                // only set if conceptname found in page
                                tmp = tmpReplace(tmp);
                                lr.setSubconcepts(tmp.replaceAll("_"," "), childCnt);
                                childCnt++;
                        }
                } // end-if
*/
                return synCnt;
        } // end getSubConcepts

        public static int getSuperConcepts(int  parentCnt, BufferedReader br, lineRecord lr ,String line, int processCnt) {

                String tmp = "";

                // check that there are no tags and the line has some text on it
```

```
            if ( (!line.contains("<") && !line.contains(">") ) && !line.trim().equals("")){
                    tmp = line.trim();
                    lr.setSuperconcepts(processCnt + "."  +tmp,parentCnt);
                    parentCnt++;
            } // end if


/*
            if (line.contains("GetRoleAndProperty.do")) {
                    int start = line.lastIndexOf("conceptname=");
                    if (start > 0) {
                            int end = line.indexOf("\"", start);
                            if (!(end > 0)) {
                                    tmp = line.substring(start + "conceptname=".length());
                            } else {
                                    tmp = line.substring(start + "conceptname=".length(), end);
                            }
                            // only set if conceptname found in page
                            tmp = tmpReplace(tmp);
                            lr.setSuperconcepts(tmp, parentCnt);
                            parentCnt++;
                    }
            } // end-if
*/
            return parentCnt;
    } // end getSuperConcepts

/*  public static void checkProcessVsNCBI(String process, lineRecord lr, int processCnt) {

            lineRecord l = (lineRecord) htbl.get("ncbi." + lr.geneName);
            StringBuffer sb = new StringBuffer();
```

```
            if (l != null) {
                for (int i = 0; i < l.getSubConceptSize() && l.getSubconcepts(i) != null; i++) {
                    if(process.equalsIgnoreCase(l.getChromosomeLocation(i)) ){
                        l.setProcesses(l.getChromosomeLocation(i)+"( replaces: " + process + ")",i);
                    }
                } // end for
            }// end if

            return;
        } // end checkConceptsVsNCBI
*/
/*      public static void addProcessesFromNCBI (String process, lineRecord lr, int processCnt) {

            lineRecord l = (lineRecord) htbl.get("ncbi." + lr.geneName);
            StringBuffer sb = new StringBuffer();

            if (l != null) {
                for (int i = 0; i < l.getSubConceptSize() && l.getSubconcepts(i,processCnt) != null; i++) {
                    if(process.equalsIgnoreCase(l.getChromosomeLocation(i)) ){
                        l.setProcesses(l.getChromosomeLocation(i)+"( replaces: " + process + ")",i);
                    }
                } // end for
            }// end if

            return;
        } // end addProcessesFromNCBI
*/
        public static String tmpReplace(String tmp){
            tmp = tmp.replace("r_s_","r's ");
            tmp = tmp.replace("t_s_","t's ");
```

```
            tmp = tmp.replace("_"," ");
            return tmp;
        } // end tmpReplace

        public static String getNCI2baseURL() {
            return NCI2baseURL;
        }

        public static void setNCI2baseURL(String nci2baseurl) {
            NCI2baseURL = nci2baseurl;
        }

} // end class DownloadHTMLFile

class lineRecord {
        String geneName = null;

        String[] diseases;

        String[] organism;

        String[] chromLoc;

        String[] processes;

        String[] subconcepts;
        String[] superconcepts;
        String[] synconcepts;

        lineRecord() {
        }
```

```
lineRecord(int d) {
        diseases        = new String[d];
        organism        = new String[d];
        chromLoc        = new String[d];
        processes       = new String[d];
        subconcepts     = new String[d];
        superconcepts = new String[d];
        synconcepts    = new String[d];
}

/**
 * @return Returns the chromosomeLocation.
 */
public String getChromosomeLocation(int i) {
        return chromLoc[i];
}

/**
 * @param chromosomeLocation
 *          The chromosomeLocation to set.
 */
public void setChromosomeLocation(String chromosomeLocation, int i) {
        this.chromLoc[i] = chromosomeLocation;
}

/**
 * @return Returns the diseases.
 */
public String getDiseases(int i) {
        return diseases[i];
```

```
}

/**
 * @param diseases
 *            The diseases to set.
 */
public void setDiseases(String diseases, int i) {
        this.diseases[i] = diseases;
}

/**
 * @return Returns the geneName.
 */
public String getGeneName() {
        return geneName;
}

/**
 * @param geneName
 *            The geneName to set.
 */
public void setGeneName(String geneName) {
        this.geneName = geneName;
}

/**
 * @return Returns the organism.
 */
public String getOrganism(int i) {
        return organism[i];
}
```

```
/**
 * @param organism
 *            The organism to set.
 */
public void setOrganism(String organism, int i) {
        this.organism[i] = organism;
}

/**
 * @return Returns the processes.
 */
public String getProcesses(int i) {
        return processes[i];
}

/**
 * @param processes
 *            The processes to set.
 */
public void setProcesses(String processes, int i) {
        this.processes[i] = processes;
}

public int getProcessSize() {
        return processes.length;
}

public int getChromLocSize() {
        return chromLoc.length;
}
```

```
public int getDiseaseSize() {
        return diseases.length;
}

public String getSubconcepts(int i) {
        return subconcepts[i];
}

public int getSubConceptSize() {
        return subconcepts.length;
}

public void setSubconcepts(String subconcepts, int i) {
        this.subconcepts[i] = subconcepts;
}

public String getSuperconcepts(int i) {
        return superconcepts[i];
}

public void setSuperconcepts(String superconcepts, int i) {
        this.superconcepts[i] = superconcepts;
}

public int getSuperConceptSize() {
        return superconcepts.length;
}

public String getSynconcepts(int i) {
        return synconcepts[i];
```

```
    }

    public void setSynconcepts(String synconcepts, int i) {
        this.synconcepts[i] = synconcepts;
    }

    public int getSynConceptSize() {
        return synconcepts.length;
    }

}
```

APPENDIX F

JAVA PROGRAM TO ANALYZE DATA

```
package ontology;

import java.io.DataInputStream;
import java.io.FileInputStream;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.Properties;
import java.util.StringTokenizer;
import java.util.Vector;

public class DataCompare {

        static Properties prop                  = null;

        private java.text.SimpleDateFormat formatter = null;

        private String destination = "";
        static logwriter geneOutput             = null;
        static logwriter chromOutput = null;
```

```
static logwriter processOutput          = null;

static  String ncbiGeneName = " ";
static  String ncbiChrom    = " ";
static  String ncbiDis      = " ";

static  String nciGeneName  = " ";
static  String nciChrom     = " ";
static  String nciDis       = " ";

static HashMap processList = null;
static HashMap synonymList = null;


static final String ADD_EXISTING   = "Addition of existing concept; ";
static final String ADD_SYNONYM          = "Addition of synonym concept; ";
static final String ADD_NEW                      = "Addition of new concept to BP and gene; ";

static final String NO_ACTION_EQUAL  = "No Action, same concepts found in both databases; ";
static final String NO_ACTION_CHILD  = "No Action, detailed concept found in NCIT; ";
static final String NO_ACTION_SYN          = "No Action, synonym concept found in NCIT; ";

static final String REPLACE_EXISTING = "Replace existing concept with child concept; ";
static final String REPLACE_SYNONYM    = "Replace concept with synonym of found concept; ";

public static void main(String[] args) {
        if(! getInputParameters(args)){
                return;
        }

        try {
```

```
FileInputStream fstream = new FileInputStream(prop.getProperty("inputFile"));
DataInputStream in = new DataInputStream(fstream);
Vector byname = new Vector();
Vector bycode = new Vector();
String tmp = "";

setupWriter();
getSynonyms(prop);

while (in.available() != 0) {
        // Print file line to screen
//      System.out.println(in.readLine());
        StringTokenizer st = new StringTokenizer(in.readLine());

        // structures used to keep found data.
        ncbiGeneName = " ";
        ncbiChrom    = " ";
        ncbiDis      = " ";
        nciGeneName  = " ";
        nciChrom     = " ";
        nciDis       = " ";

        Vector ncbi = new Vector();
        Vector nci  = new Vector();
        Vector subConcepts = new Vector();
        Vector superConcepts = new Vector();
        Vector synConcepts = new Vector();

        StringBuffer sb = new StringBuffer();
        int tokens = 0;
        int tokenCounter = 0;
```

```
while(st.hasMoreElements()){
        tokenCounter = st.countTokens();
        tmp = st.nextToken(";").toLowerCase().trim();
        tokens++;
        String process = "";

        if (! tmp.equals("")){
                switch (tokens){
                case 1: ncbiGeneName = tmp ; break;
                case 2: ncbiChrom = tmp ; break;
                case 3:
                        if ( (tmp != null) && (! tmp.equals("")) ){
                                StringTokenizer stTmp = new StringTokenizer(tmp);
                                while (stTmp.hasMoreElements()){
                                        int i = stTmp.countTokens();
                                        process = stTmp.nextToken(",").toLowerCase().trim();
                                        ncbi.add(process);
                                }
                        }// end if
                        break;
                case 4: ncbiDis = tmp ; break;
                case 5: nciGeneName = tmp ; break;
                case 6: nciChrom = tmp ; break;
                case 7:
                        if ( (tmp != null) && (! tmp.equals("")) ){
                                StringTokenizer stTmp = new StringTokenizer(tmp);
                                while (stTmp.hasMoreElements()){
                                        int i = stTmp.countTokens();
                                        process = stTmp.nextToken(",").toLowerCase().trim();
                                        nci.add(process);
                                }
```

```
        }// end if
        break;
case 8: nciDis= tmp ; break;
case 9:
        if ( (tmp != null) && (! tmp.equals("")) ){
                StringTokenizer stTmp = new StringTokenizer(tmp);
                while (stTmp.hasMoreElements()){
                        int i = stTmp.countTokens();
                        process = stTmp.nextToken(",").toLowerCase().trim();
                        superConcepts.add(process);
                }
        }// end if
        break;
case 10:
        if ( (tmp != null) && (! tmp.equals("")) ){
                StringTokenizer stTmp = new StringTokenizer(tmp);
                while (stTmp.hasMoreElements()){
                        int i = stTmp.countTokens();
                        process = stTmp.nextToken(",").toLowerCase().trim();
                        subConcepts.add(process);
                }
        }// end if
        break;
case 11:
        if ( (tmp != null) && (! tmp.equals("")) ){
                StringTokenizer stTmp = new StringTokenizer(tmp);
                while (stTmp.hasMoreElements()){
                        int i = stTmp.countTokens();
                        process = stTmp.nextToken(",").toLowerCase().trim();
                        synConcepts.add(process);
                }
```

```
                                        }// end if
                                        break;

                                    default: System.out.println("Did not find anything here for int=" + tokens); break;
                                    } // end switch
                                } // end if
                            } // end while
//                          sb = sortVectors(st,sb,ncbi,nci);
                            sb = runDataCompare2(st,sb,ncbi,nci,subConcepts,superConcepts,synConcepts);
//                          output.log(sb);
                            chromOutput.log(ncbiGeneName + "; " + ncbiChrom + "; " + nciChrom + "; " );

                    } // end while
                    System.out.println(bycode.toString());
                    in.close();
            } // end try
            catch (Exception e){
                    e.printStackTrace();
            }
    } // end Main

    public static boolean getInputParameters(String[] args) {
            try {
                    // Open the file that is the first
                    // command line parameter
                    if (args[0].equalsIgnoreCase("-h")){
                            System.out.println("-p PathToPropertyFile");
                            return false;
                    }
```

```
                if (args[0].equalsIgnoreCase("-p")){
                        prop = new Properties();
                        try{
                                FileInputStream fis = new FileInputStream(args[1]);
                                prop.load(fis);
                                System.out.println(prop.toString());
                        }
                        catch (Exception e) {
                                System.err.println("Could not open properties file -p filename");
                                return false;
                        }
                }
        }
        catch (Exception e) {
                System.err.println("Could not open properties file -p filename");
                return false;
        }
        return true;
} // end getInputParameters

public static void setupWriter(){
        geneOutput = new logwriter();
        geneOutput.init(prop.getProperty("outputDir"),prop.getProperty("geneOutputFileName"));
        StringBuffer header = new StringBuffer();
        header.append("Gene Name; NCBI Process; Status;  NCI Process; Resultant Process ");
        geneOutput.log(header);

        chromOutput = new logwriter();
        chromOutput.init(prop.getProperty("outputDir"),prop.getProperty("chromOutputFileName"));
        header = new StringBuffer();
        header.append("Gene Name; NCBI Chromosome Location; NCI Chromosome Location ");
```

```
        chromOutput.log(header);

        processOutput = new logwriter();
        processOutput.init(prop.getProperty("outputDir"),prop.getProperty("processOutputFileName"));
        header = new StringBuffer();
        header.append("Gene Name; NCBI Process list; NCI Process list ");
        processOutput.log(header);

}

public static StringBuffer sortVectors(StringTokenizer st, StringBuffer sb, Vector ncbi, Vector nci){
        StringBuffer returnSB = null;
        ResultSet nciRS = null;
        ResultSet ncbiRS = null;
        Vector found = new Vector();
        String qry = "Select * from ass,processes where ass.id = processes.id && processes.string = \"";

        Enumeration eNCBI = ncbi.elements();
        Enumeration eNCI = nci.elements();

        // first compare the 2 vectors for like strings
        int i = 0;
        boolean reset = false;
        while (eNCBI.hasMoreElements()){
                if (reset) {
                        i=0;
                        reset=false;
                }

                String ncbiTmp = (String) eNCBI.nextElement();
                int j=0;
```

```
                eNCI = nci.elements();
                while (eNCI.hasMoreElements()){
                        String nciTMP = (String)eNCI.nextElement();
                        if (ncbiTmp.equals(nciTMP) ) {
                                ncbi.remove(i);
                                nci.remove(j);
                                found.add(ncbiTmp + ",");
                                eNCBI = ncbi.elements();
                                reset=true;
                                break;
                        } // end if
                        j++;
                } // end while
                i++;
        } // end while

        eNCBI = ncbi.elements();
        reset=false;
        // loop thru each ncbi element
        while (eNCBI.hasMoreElements()){
/*              if (reset) {
                        i=0;
                        reset=false;
                }
*/
                // ncbiTmp is the specific ncbi process string
                String ncbiTmp = (String) eNCBI.nextElement();

//              ncbiRS = executeInquiry(qry + ncbiTmp + "\"");
//              System.out.println(qry + ncbiTmp + "\"");
                // does it have associations?
```

```
if(ncbiRS != null){
        try {
                i = 0;
//              ncbiRS.next();
                while(ncbiRS.next()){
                        //String rsString = ncbiRS.getString("string");
                        // rsLink is the id of an associated process
                        String rsLink   = ncbiRS.getString("link");
//                      nciRS = executeInquiry("select * from processes where id = \"" + rsLink + "\"");
                        // System.out.println("select * from processes where id = \"" + rsLink + "\"");
                        //setup the result set
//                      nciRS.next();
                        while(nciRS.next()){
                                String nciString = nciRS.getString("string");
                                int j=0;
                                eNCI = nci.elements();
                                while (eNCI.hasMoreElements()){
                                        String nciTMP = (String)eNCI.nextElement();
                                        if (nciString.equals(nciTMP) ) {
                                                ncbi.remove(i);
                                                nci.remove(j);
                                                found.add(ncbiTmp + "(" + nciTMP+ "),");
//                                              eNCBI = ncbi.elements();
//                                              reset=true;
//                                              break;
                                        } // end if
                                        j++;
                                } // end while
                        } // end while
                } // end while
        } // end try
```

```
                    catch (SQLException sqle){
                        sqle.printStackTrace();
                        return sb;
//                  break;
                    }
                }//end if
            }

            eNCI = nci.elements();

            // loop thru each ncbi element
            while (eNCI.hasMoreElements()){
                // ncbiTmp is the specific ncbi process string
                String nciTmp = (String) eNCI.nextElement();

//              nciRS = executeInquiry(qry + nciTmp + "\"");
                // does it have associations?
                if(nciRS != null){
                    try {
                        i = 0;
                        while(nciRS.next()){
                            // rsLink is the id of an associated process
                            String rsLink   = nciRS.getString("link");
//                          ncbiRS = executeInquiry("select * from processes where id = \"" + rsLink + "\"");
                            //setup the result set
                            while(ncbiRS.next()){
                                String ncbiString = ncbiRS.getString("string");
                                int j=0;
                                eNCBI = ncbi.elements();
                                while (eNCBI.hasMoreElements()){
                                    String ncbiTmp = (String)eNCBI.nextElement();
```

```
                                                    if (ncbiString.equals(ncbiTmp) ) {
                                                            ncbi.remove(j);
                                                            nci.remove(i);
                                                            found.add(ncbiTmp + "(" + nciTmp+ "),");
                                                    } // end if
                                                    j++;
                                            } // end while
                                    } // end while
                            } // end while
                    } // end try
                    catch (SQLException sqle){
                            sqle.printStackTrace();
                            return sb;
//                  break;
                    }
            }//end if

        }

        return stTosb(ncbi,nci,found);
//      return returnSB;
    }

    public static StringBuffer stTosb(Vector ncbi,Vector nci,Vector found){

        return stTosb(ncbi,nci,found, null);
    }

    public static StringBuffer stTosb(Vector ncbi,Vector nci,Vector found, Vector replaced){
        return stTosb(ncbi,nci,found, null, null);
```

```
}

public static StringBuffer stTosb(Vector ncbi,Vector nci,Vector found, Vector replaced, Vector add){
        String tmp = null;
        int i = 0;
        StringBuffer returnSB = new StringBuffer();
        Enumeration Enumer = null;


        returnSB.append(ncbiGeneName);
        returnSB.append(" ;");
        returnSB.append(ncbiChrom);
        returnSB.append(" ;");
        // rewrite ncbi processes
        if (ncbi.size() > 0){
                Enumer = ncbi.elements();
                while (Enumer.hasMoreElements()){
                        returnSB.append( (String)Enumer.nextElement() + ",");
                } // end while
        }
        returnSB.append(" ;");

        returnSB.append(ncbiDis);
        returnSB.append(" ;");

        // nci fields
        returnSB.append(nciGeneName);
        returnSB.append(" ;");
        returnSB.append(nciChrom);
        returnSB.append(" ;");
```

```
if(nci.size() > 0){
        Enumer = nci.elements();
        while (Enumer.hasMoreElements()){
                returnSB.append( (String)Enumer.nextElement() + ",");
        } // end while
}
returnSB.append(" ;");

returnSB.append(nciDis);
returnSB.append(" ;");
/*      // add like terms to sb from Vector found
if (found.size() > 0){
        Enumer = found.elements();
        while (Enumer.hasMoreElements()){
                returnSB.append( (String)Enumer.nextElement());
        } // end while
}
returnSB.append(" ;");
//      returnSB.append();
*/
// add like terms to sb from Vector replaced
if (replaced != null && (replaced.size() > 0)) {
        Enumer = replaced.elements();
        if(replaced.size() > 0){
                Enumer = replaced.elements();
                while (Enumer.hasMoreElements()){
                        returnSB.append( (String)Enumer.nextElement() + ",");
                } // end while
        }
}
returnSB.append(" ;");
```

```
        // add like terms to sb from Vector replaced
        if (add != null && (add.size() > 0)) {
                Enumer = add.elements();
                if(add.size() > 0){
                        Enumer = add.elements();
                        while (Enumer.hasMoreElements()){
                                returnSB.append( (String)Enumer.nextElement() + ",");
                        } // end while
                }
        }
        returnSB.append(" ;");

        return returnSB;
} //end stTosb


public static StringBuffer processStToSb(Vector ncbi,Vector nci){
        String tmp = null;
        int i = 0;
        StringBuffer returnSB = new StringBuffer();
        Enumeration Enumer = null;


        // rewrite ncbi processes
        if (ncbi.size() > 0){
                Enumer = ncbi.elements();
                while (Enumer.hasMoreElements()){
                        returnSB.append( (String)Enumer.nextElement() + ",");
                } // end while
        }
```

```
            returnSB.append(" ;");

            if(nci.size() > 0){
                    Enumer = nci.elements();
                    while (Enumer.hasMoreElements()){
                            returnSB.append( (String)Enumer.nextElement() + ",");
                    } // end while
            }
            returnSB.append(" ;");


            return returnSB;
    } //end stTosb

    public static boolean stringComp(String ncbiProcess, String ncitProcess){
            boolean state = false;

            int pos = ncitProcess.indexOf(".");

            String tmp = ncitProcess.substring(pos);

            if (ncbiProcess.equalsIgnoreCase(tmp)) state=true;


            return state;

    }

    public static StringBuffer runDataCompare(StringTokenizer st, StringBuffer sb, Vector ncbi, Vector nci,Vector
superConcepts, Vector subConcepts, Vector synConcepts){
            StringBuffer returnSB = null;
```

```
ResultSet nciRS = null;
ResultSet ncbiRS = null;

Vector found = new Vector();
Vector replaced = new Vector();
Vector add = new Vector();

Enumeration eNCBI = ncbi.elements();
Enumeration eNCI = nci.elements();
Enumeration tmpEnum = nci.elements();

int i = 0;
boolean reset = false;
boolean located = false;

//  [ "equal" ]
// first compare the 2 vectors for like strings
// always add to "replaced" Vector to allow comparison at end


if (ncbi.size() <= 0){
        return stTosb(ncbi,nci,found, replaced);
}

while (eNCBI.hasMoreElements()){
        if (reset) {
                i=0;
                reset=false;
        }

        // get each NCBI process
```

```
                String ncbiTmp = (String) eNCBI.nextElement();
                eNCI = nci.elements();
                while (eNCI.hasMoreElements()){
                        String nciTMP = (String)eNCI.nextElement();
//                      int spot = ncbiTmp.indexOf(".");
//                      String number = ncbiTmp.substring(spot);
                        if (ncbiTmp.equals(nciTMP) ) {
                                // remove it from ncbi Vector since it's found
                                ncbi.remove(i);
//                              replaced.add(nciTMP + ",");
                                eNCBI = ncbi.elements();
                                located = true;
                                reset=true;
                                break;
                        } // end if
                } // end while
                i++;
        } // end while

        eNCBI = ncbi.elements();
        reset=true;

        // [ "children equal" ]
        // replace NCI with child

        while (eNCBI.hasMoreElements()){
                if (reset) {
                        i=0;
                        reset=false;
                }
```

```
        String ncbiTmp = (String) eNCBI.nextElement();
        int location =0;
        tmpEnum = subConcepts.elements();
        while (tmpEnum.hasMoreElements()){
                String nciTMP = (String)tmpEnum.nextElement();

                int spot = nciTMP.indexOf(".");
                if(spot>-1){
                        location = Integer.parseInt(nciTMP.substring(0,spot));
                }
                nciTMP = nciTMP.substring(spot+1);

                if (ncbiTmp.equals(nciTMP) ) {
                        // remove it from ncbi Vector
                        ncbi.remove(i);
                        eNCBI = ncbi.elements();
                        replaced.add(ncbiTmp +"[ replacing: " + nci.get(location)+ "]" + "," );
                        located = true;
                        reset=true;
                        break;
                } // end if
        } // end while
        i++;
} // end while

eNCBI = ncbi.elements();
reset=true;

//  [ "children's synonym equal" ]
// replace NCI with synonym of child
```

```
while (eNCBI.hasMoreElements()){
        if (reset) {
                i=0;
                reset=false;
        }

        String ncbiTmp = (String) eNCBI.nextElement();
        int location = 0;
        tmpEnum = synConcepts.elements();
        while (tmpEnum.hasMoreElements()){
                String nciTMP = (String)tmpEnum.nextElement();
                int spot = nciTMP.indexOf(".");
                try {
                        if(spot>-1){
                                location = Integer.parseInt(nciTMP.substring(0,spot));
                        }
                }
                catch (Exception e){
                        e.printStackTrace();

                }
                nciTMP = nciTMP.substring(spot+1);
                if (ncbiTmp.equals(nciTMP) ) {
                        // remove it from ncbi Vector
                        ncbi.remove(i);
                        eNCBI = ncbi.elements();
                        replaced.add(ncbiTmp + "[ replacing: " + nci.get(location)+ "]" + ",");
                        located = true;
                        reset=true;
                        break;
                } // end if
```

```
                } // end while
                i++;
            } // end while

            returnSB = stTosb(ncbi,nci,found, replaced);

            return returnSB;
        }

        // most recent data comparison mechanism written 3/28/06
        public static StringBuffer runDataCompare2(StringTokenizer st, StringBuffer sb, Vector ncbi, Vector nci,Vector
superConcepts, Vector subConcepts, Vector synConcepts){
            StringBuffer returnSB = null;
            ResultSet nciRS = null;
            ResultSet ncbiRS = null;

            Vector found   = new Vector();
            Vector replaced = new Vector();
            Vector add          = new Vector();
            Vector ncbiCopy     = new Vector();

            Enumeration eNCBI = ncbi.elements();
            Enumeration eNCI = nci.elements();
            Enumeration tmpEnum = nci.elements();

            int i = 0;
            boolean reset = false;
            boolean located = false;

            //  [ "equal" ]
            // first compare the 2 vectors for like strings
```

```
// always add to "replaced" Vector to allow comparison at end


if (ncbi.size() <= 0){
        return stTosb(ncbi,nci,found, replaced);
}

// keep a copy of original ncbi processes
while (eNCBI.hasMoreElements()){
        // get each NCBI process
        String ncbiTmp = (String) eNCBI.nextElement();
        ncbiCopy.add(ncbiTmp);
}

eNCBI = ncbi.elements();

// run thru this loop looking for like strings in ncbi and ncit
while (eNCBI.hasMoreElements()){
        if (reset) {
                i=0;
                reset=false;
        }

        // get each NCBI process
        String ncbiTmp = (String) eNCBI.nextElement();
        eNCI = nci.elements();
        while (eNCI.hasMoreElements()){
                String nciTMP = (String)eNCI.nextElement();
//              int spot = ncbiTmp.indexOf(".");
//              String number = ncbiTmp.substring(spot);
                if (ncbiTmp.equals(nciTMP) ) {
```

```
                                        // remove it from ncbi Vector since it's found
                                        ncbi.remove(i);
//                                      replaced.add(nciTMP + ",");
                                        eNCBI = ncbi.elements();
                                        located = true;
                                        reset=true;
                                        geneOutput.log(ncbiGeneName + "; " + ncbiTmp + "; " + NO_ACTION_EQUAL + nciTMP+ ";
" );
                                        break;
                                } // end if
                        } // end while
                        i++;
                } // end while

                eNCBI = ncbi.elements();
                reset=true;


                // at this point, ncbi vector contains all processes that did not string match
                // further logic required to find a "like" nci process

                //  [ "children equal" ]
                // replace NCI with child

                // check for each ncbi process in processList hashmap and see if it exists

                // run thru this loop looking checking if ncbi processes are in ncit process heirarchy
                while (eNCBI.hasMoreElements()){
                        if (reset) {
                                i=0;
                                reset=false;
```

```
                }

                // get each NCBI process
                String ncbiTmp = (String) eNCBI.nextElement();
                Process p = (Process)processList.get(ncbiTmp);

                if (p != null){
                        // if we get here, process is in BP
                        // we're now searching if parent of process
                        eNCI = nci.elements();
                        while (eNCI.hasMoreElements()){
                                String nciTMP = (String)eNCI.nextElement();
                                // reset p each time we get another NCI process
                                p = (Process)processList.get(ncbiTmp);

                                // check if nciTMP = ncbi concept

                                if (p.getConceptName().equalsIgnoreCase(nciTMP)){
                                        ncbi.remove(i);
                                        replaced.add(ncbiTmp +"[ equals NCI concept: " + nciTMP + " in the NCBI]" + "," );
                                        eNCBI = ncbi.elements();
                                        located = true;
                                        reset=true;
                                        geneOutput.log(ncbiGeneName + "; " + ncbiTmp + "; " + NO_ACTION_EQUAL +
nciTMP+ "; " );
                                }
                                else{
                                        // loop thru parents in BP
                                        while (p != null){
                                                String parent = p.getParentName();
                                                if (nciTMP.equalsIgnoreCase(parent) ) {
```

```
                                                    //  remove it from ncbi Vector since it's found
                                                    ncbi.remove(i);
                                                    replaced.add(ncbiTmp +"[ should replace NCI concept: " + nciTMP + "
because a child concept of this process is found in the NCBI]" + "," );
                                                    eNCBI = ncbi.elements();
                                                    located = true;
                                                    reset=true;
                                                    geneOutput.log(ncbiGeneName + "; " + ncbiTmp + "; " +
REPLACE_EXISTING + nciTMP + "; " + ncbiTmp + " ; ");
                                                    break;
                                            } // end if (ncbiTmp.equals(nciTMP))

                                            // get parent process for this process
                                            p = (Process)processList.get(parent);

                                    } // end while (String parent = p.getParentName())

                                    // here, found ncbiTmp in process hierarchy but didn't find anything in nci process list

                                    if (! reset){
                                            ncbi.remove(i);
                                            eNCBI = ncbi.elements();
                                            located = true;
                                            reset=true;
                                            geneOutput.log(ncbiGeneName + "; " + ncbiTmp + "; " + ADD_EXISTING + " "
+ "; " + ncbiTmp + "; ");

                                            break;
                                    }
                            } // end if (p.getConceptName().equalsIgnoreCase(nciTMP)){

                    } // end while (eNCI.hasMoreElements())
```

```
        } // end if (p != null)
        i++;
} // end while

eNCBI = ncbi.elements();
reset=true;


// at this point, ncbi vector does not contain any processes matching processes in NCI
// now, we'll check if synonyms exist for ncbi processes in nci process hierarchy

//  [ "synonym found" ]
// replace NCI if it's a child of process

// check for each ncbi process in processList hashmap and see if it exists

// run thru this loop looking checking if ncbi processes are in ncit process heirarchy as synonyms
while (eNCBI.hasMoreElements()){
        if (reset) {
                i=0;
                reset=false;
        }

        // get each NCBI process
        String ncbiTmp = (String) eNCBI.nextElement();
        Process p = (Process)synonymList.get(ncbiTmp);

        if (p != null){
                // if we get here, process is in BP
                // we're now searching if parent of process
                eNCI = nci.elements();
```

```
while (eNCI.hasMoreElements()){
        String nciTMP = (String)eNCI.nextElement();
        // reset p each time we get another NCI process
        p = (Process)synonymList.get(ncbiTmp);
        String conceptOfSyn = p.getConceptName();

        if (p.getConceptName().equalsIgnoreCase(nciTMP)){
                ncbi.remove(i);
                replaced.add(ncbiTmp +"[ equals NCI concept: " + nciTMP + " in the NCBI]" + "," );
                eNCBI = ncbi.elements();
                located = true;
                reset=true;
                geneOutput.log(ncbiGeneName + "; " + ncbiTmp + "; " + NO_ACTION_SYN +
nciTMP+ "; " );
        }
        else{
                // loop thru parents in BP
                while (p != null){
                        String parent = p.getParentName();
                        if (nciTMP.equalsIgnoreCase(parent) ) {
                                //  remove it from ncbi Vector since it's found
                                ncbi.remove(i);
                                replaced.add(ncbiTmp +"[ should replace NCI concept: " + nciTMP + "
via a synonym found in NCBI for concept: " + p.getParentName() + "]" + "," );
                                eNCBI = ncbi.elements();
                                located = true;
                                reset=true;
                                geneOutput.log(ncbiGeneName + "; " + ncbiTmp + "; " +
REPLACE_SYNONYM + nciTMP+ "; " + p.getConceptName() + "; ");
                                break;
                        } // end if (ncbiTmp.equals(nciTMP))
```

```
                                        // get parent process for this process
                                        p = (Process)processList.get(parent);

                                    } // end while (String parent = p.getParentName())
                                    // here, found ncbiTmp in process hierarchy but didn't find anything in nci process list

                                    if (! reset){
                                        ncbi.remove(i);
                                        eNCBI = ncbi.elements();
                                        located = true;
                                        reset=true;
                                        geneOutput.log(ncbiGeneName + "; " + ncbiTmp + "; " + ADD_SYNONYM + "
; " + conceptOfSyn + "; ");

                                        break;
                                    }

                                } // end if (p.getConceptName().equalsIgnoreCase(nciTMP))

                            } // end while (eNCI.hasMoreElements())
                        } // end if (p != null)
                        i++;
                } // end while

                eNCBI = ncbi.elements();
                reset=true;


                // at this point, all remaining entities in NCBI vector need to be added to NCI.
                // run thru this loop looking checking if ncbi processes are in ncit process heirarchy as synonyms
                while (eNCBI.hasMoreElements()){
```

```
                    if (reset) {
                            i=0;
                            reset=false;
                    }

                    // get each NCBI process
                    String ncbiTmp = (String) eNCBI.nextElement();
                    geneOutput.log(ncbiGeneName + "; " + ncbiTmp + "; " + ADD_NEW + " " + "; " + ncbiTmp + "; " );
            } // end while

/*      commented this code b/c it's old logic for finding synonyms.  Replaced with processList logic.
 *
 *      while (eNCBI.hasMoreElements()){
                    if (reset) {
                            i=0;
                            reset=false;
                    }

                    String ncbiTmp = (String) eNCBI.nextElement();
                    int location =0;
                    tmpEnum = subConcepts.elements();
                    while (tmpEnum.hasMoreElements()){
                            String nciTMP = (String)tmpEnum.nextElement();

                            int spot = nciTMP.indexOf(".");
                            if(spot>-1){
                                    location = Integer.parseInt(nciTMP.substring(0,spot));
                            }
                            nciTMP = nciTMP.substring(spot+1);

                            if (ncbiTmp.equals(nciTMP) ) {
```

```
                        // remove it from ncbi Vector
                        ncbi.remove(i);
                        eNCBI = ncbi.elements();
                        replaced.add(ncbiTmp +"[ replacing: " + nci.get(location)+ "]" + "," );
                        located = true;
                        reset=true;
                        break;
                } // end if
        } // end while
        i++;
} // end while


eNCBI = ncbi.elements();
reset=true;

//  [ "children's synonym equal" ]
// replace NCI with synonym of child

while (eNCBI.hasMoreElements()){
        if (reset) {
                i=0;
                reset=false;
        }

        String ncbiTmp = (String) eNCBI.nextElement();
        int location = 0;
        tmpEnum = synConcepts.elements();
        while (tmpEnum.hasMoreElements()){
                String nciTMP = (String)tmpEnum.nextElement();
                int spot = nciTMP.indexOf(".");
```

```
                    try {
                        if(spot>-1){
                            location = Integer.parseInt(nciTMP.substring(0,spot));
                        }
                    }
                    catch (Exception e){
                        e.printStackTrace();

                    }
                    nciTMP = nciTMP.substring(spot+1);
                    if (ncbiTmp.equals(nciTMP) ) {
                        // remove it from ncbi Vector
                        ncbi.remove(i);
                        eNCBI = ncbi.elements();
                        replaced.add(ncbiTmp + "[ replacing: " + nci.get(location)+ "]" + ",");
                        located = true;
                        reset=true;
                        break;
                    } // end if
                } // end while
                i++;
            } // end while
*/
            returnSB = stTosb(ncbiCopy,nci,found, replaced, ncbi);

            sb = processStToSb(ncbiCopy, nci);
            processOutput.log(ncbiGeneName + "; " + sb);

            return returnSB;
        }
```

```
public static void getSynonyms(Properties prop){
        try {
                FileInputStream fstream = new FileInputStream(prop.getProperty("synonymFile"));
                DataInputStream in = new DataInputStream(fstream);
                processList = new HashMap();
                synonymList = new HashMap();
                while (in.available() != 0) {
                        // Print file line to screen
//                      System.out.println(in.readLine());
                        Process process = new Process();

                        StringTokenizer st = new StringTokenizer(in.readLine());

                        StringBuffer sb = new StringBuffer();
                        int tokens = 0;
                        int tokenCounter = 0;
                        String tmp = "";
                                while(st.hasMoreElements()){
                                        tokenCounter = st.countTokens();
                                        tmp = st.nextToken(",").toLowerCase().trim();
                                        tokens++;

                                        if (! tmp.equals("")){
                                                switch (tokens){
                                                case 1: process.setConceptID(tmp);      break;
                                                case 2: process.setConceptName(tmp);    break;
                                                case 3: process.setParentName(tmp);     break;
                                                case 4: process.setSynonym(tmp);        break;
                                                case 5: process.setFlag(tmp);           break;
                                                } // end switch
```

REFERENCES

1. A. Kumar and B. Smith. (2003). The Unified Medical Language System and the Gene Ontology: Some critical reflections. *Advances in Artificial Intelligence* (Lecture Notes in Artificial Intelligence 2821), 135-148, Berlin: Springer, 2003.

2. B. Smith, J. Williams, and S. Schulze-Kremer. (2003). The Ontology of the Gene Ontology. *2003 AMIA Annual Symposium*, 609-613.

3. W. Ceusters, B. Smith, C. Kumar, and C. Dhaen. (2003). Mistakes in medical ontologies: Where do they come from and how can they be detected? *Ontologies in Medicine: Proc. Workshop on Medical Ontologies*, 145-164.

4. W. Ceusters, B. Smith, and L. Goldberg. (2005). A terminological and ontological analysis of the NCI Thesaurus. *Methods of Information in Medicine*, (44), 498-507.

5. H. Min, Y. Perl, Y. Chen, M. Halper, J. Geller, and Y. Wang. Auditing as part of the terminology design life cycle. Submitted for journal publication.

6. *caCORE technical guide*. Retrieved November 1, 2005 from the NIH web site: http://www.ncicb.nci.nih.gov.

7. R. J. Brachman and H. J. Levesque. (1984). The tractability of subsumption in frame based description languages. *AAAI*, (84), 34-37.

8. R. J. Brachman and J. Schmolze. (1985). An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2), 171-216.

9. D. Nardi and R. J. Brachman. (2002). An introduction to description logics. *The Description Logic Handbook*, 5-44.

10. F. S. Collins, A. Patrinos, E. Jordan, A. Chakravarti, R. Gesteland, L. Walters, et al.(1998). New goals for the U.S. human genome project: 1998.2003. *Science*, (282), 682-689.

11. J. H. Lin. (1998). Divining and altering the future: Implications from the human genome project. *Science*, (282), 1532.

12. Z. E. Karanjawala and F. S. Collins.(1998). Genetics in the context of medical practice. *JAMA*, 280(17), 1533-1534, 1998.

13. G.O. Consortium. Creating the Gene Ontology resource: Design and Implementation. Genome Res. (11), 1425-1433, 2001.

14. J. Lomax and A. T. McCray. (2004). Mapping the Gene Ontology into the Unified Medical Language System. *Comparative and Functional Genomics*. 5(5), 354-361.

15. *Java Technology*. Retrieved October 20, 2005 from the Sun web site: http://java.sun.com.

16. *Marc Oren's NJIT Thesis documents*. Retrieved April 17, 2006 from the NJIT web site: http://web.njit.edu/~mo24.