1-1-2007

Meta search engine

Kwok-Pun Chan
Ryerson University

META SEARCH ENGINE

By

Kwok-Pun Chan

A project presented to Ryerson University

in partial fulfillment of the requirements for the degree of

Master of Engineering

in the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2007

I hereby declare that I am the sole author of this project.

I authorize Ryerson University to lend this project to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this project by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Meta search engine: To allow multiple engine searches on both window and Linux platforms. The meta search engine reduces biased information and improves quality of search results. It also supports sorting results in an alphabetic or relevancy order.

Kwok-Pun Chan

Master of Engineering, Ryerson University 2007

ABSTRACT

Meta search engines allow multiple engine searches to minimize biased information and improve the quality of the results it generates. However, existing meta engine applications contain many foreign language results, and only run on window platform. The meta search engine we develop will resolve these problems. Our search engine will run on both window and Linux platforms, and has some desirable properties: 1) users can shorten the search waiting time if one of the search engines is down; 2) users can sort the result titles in an alphabetic or relevancy order. Current meta search website only allows users to sort results by relevancy only. Our search engine allows users to do an alphabetical search from the previous relevancy search result, so that the users can identify the required title within a shorter timeframe.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

The rapid growth of internet resources has led to a huge amount of available information. As the amount of information continues to grow, so does the complexity of finding and retrieving it. The constant and continuous changing of web pages nature also make searching the Web a challenging task.

Currently, all search engines have put a lot of effort into developing better searching and ranking algorithm. Unfortunately, different search engines possess a number of different deficiencies [Table 1] which could be poor precision, low coverage of the web, out of date databases, a large number of dead links, inconsistent user interfaces, or difficulties with spamming.

Search Engine	Of Dead Links %	Of Type-400 Errors %
AltaVista	13.7%	9.3%
Google	4.3%	3.3%
HotBot	2.3%	2.0%
MSN	1.7%	1.0%

Table 1. Percentage of dead links and type-400 errors occurring in the query results of selected search engines (data from February, 2000).

Meta search engines have thus been introduced to overcome some of these difficulties. It collects results by querying a selection of search engines in parallel, and displays an integrated result list to users in a uniform format. It thus solves some user problems like which search engine to use and when to use them. It also improves the coverage of the entire web. Because of the giant indexes, most of the search engines will return thousands or millions of relevant documents for a particular query, but only a few of which might be useful to a particular user. Also, since different search engines use

different ranking algorithms, one search engine might not give the information that user really wants. So by doing a combine search, the users will have a better chance to get the useful information that they actually require.

A combined search on different search engines can minimize each search engine's drawbacks, disadvantages and their skew search results caused by webmasters around the world who constantly try to find ways to boost up their ranking. They might want to alter their site so that it will custom fit to that particular search engine algorithm. For example, some of web builders will use "search engine spamming" whereby they add possibly unrelated keywords to their pages in order to alter the ranking of their pages. Therefore, a combined search will allow users to have a less biased search results from different search engines' sorting and ranking algorithms.

To see how meta search can lead to improved results, consider how electrical engineers perform averaging of noisy signals. The averaging cancels out random noise and reveals the original noise free signal. Since web noise affects regular search engines in different ways, meta search filters noise by averaging the votes of the underlying engines, resulting better results. For meta search engines to live up to this promise, they must combine search results by weighted voting, not by round robin and they should use the most popular underlying search engines.

Dreilinger and Howe [16] suggested that a meta search engine must contain three components: a dispatch mechanism, interface agents, and a display mechanism. Figure 1 illustrates the organization of these three components. The dispatcher is the algorithm or decision making approach used to determine which search engines a specific query should be sent to. The success of meta searching depends critically on carefully

selecting which resources to use as well as the size, the content, the number of search engines, and how well a meta search engine selects individual search engines that would most likely return the best results for a particular query. The interface agents are self contained programs that manage the interaction with a particular search engine. They are responsible for translating the user query format into the format of a particular search engine because the format of the queries and the ways search engines process them are all different. The interface agents are also responsible for interpreting the different results format of each engine. The display mechanism combines the raw results returned from each search engine, removes duplicates, and displays them to the users. This is necessary because results should ideally be displayed in a uniform format and be ordered by rank.

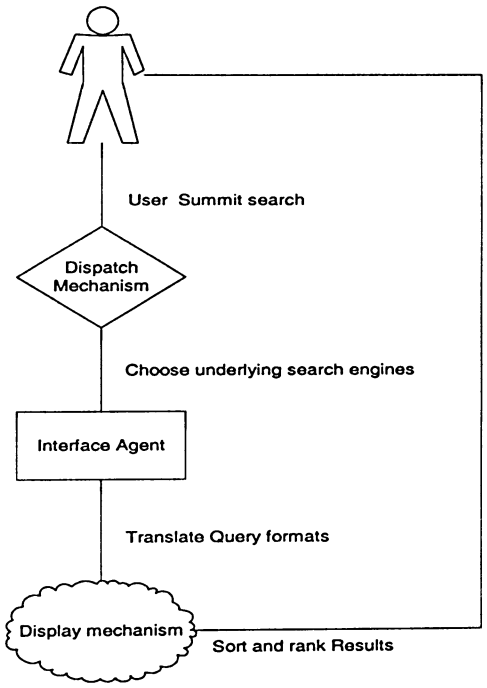

Fig 1. Dreilinger & Howe MSE Model

In this project, we developed a meta search engine which dispatches user queries to multiple search services and displays an integrated results list back to them. Our meta search engine is composed of four search engines: Google [1], Lycos [2], Hotbot [3] and AltaVista [4]. It is implemented with Java technologies and can work on multiple platforms.

Our meta search engine consists of the three components as suggested by Dreilinger and Howe's [16]. These components have the following addition characteristics:

1) **Dispatch mechanism** - Instead of using a decision making algorithm ,our meta search engine gives user a few options to choose like "searching time out", "number of search engines used" and "search engines combination" options.

2) **Interface agent** - We used regexp java regular expression package to perform the URL extraction on four different major underlying search engines.

3) **Display mechanism** - We assigned numeric rank to every hit on individual engine. When we merge results from two or more engines, combined hit receives rank which is the average of all engines rank minus the number of engines that give the hit. So the multi-engine hits receive a higher priority.

In addition, our search engine has extra features: it can work in Window and Linux, and it also can give user a choice of sorting the results alphabetically. In the following sections, we will resume each search engine background, as well as our meta search engine's features and all improvements over some of the other meta search engines which can currently be purchased or downloaded for a free trial. After that, we will talk about the implementation of our program with some experimental screen shots. The final section gives the conclusion of the project.

2.0 BACKGROUND

We first focus on Google [1] because it is the most popular search engine these days. The use of three search engines like Lycos [2], Hotbot [3] and AltaVista [4] will cover the three other major underlying search engines like Ask [5], Msn [6] and Yahoo [7]. As a result, our meta search engine covers four of the major current search engines.

2.1 GOOGLE

Google [1] uses an algorithm called PageRank [8] to rank web pages that match a given search string. It interprets a link from page A to page B as a vote by page A for page B. But, it looks for more than just the number of votes, or links a page received. It also analyzes the page that casts the vote. Votes casted by pages that are themselves weighed more heavily will also be weighted more. In other words, a hyperlink to a page counts as a vote of support; and a page that is linked to by many pages with higher PageRank receives a higher rank itself. If there are no links to a web page, there is no support for that page. PageRank is a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page. PageRank can be calculated for any size collection of documents. It is assumed in several research papers that the distribution is evenly divided between all documents in the collection at the beginning of the computational process. The PageRank computations requires several iterations through the collection to adjust approximate PageRank values to more closely reflect the theoretical true value. A probability is expressed as a numeric value between 0 and 1. Hence, a PageRank of 0.5 means there is a 50% chance that a person clicking on a random link will be directed to the document.

Suppose a small universe of four web pages: A, B, C and D. The initial approximation of PageRank would be evenly divided between these four documents. Hence, each document would begin with an estimated PageRank of 0.25. If each of the pages B, C, and D only links to A; they would each contributes 0.25 PageRank to A. Let PR(X) be the PageRank of X, then PR (A) can be expressed as:

$$PR(A) = PR(B) + PR(C) + PR(D).$$

Now, let suppose page B also has a link to page C, and page D has links to all three pages. The value of the link-votes is divided among all the outbound links on a page. Thus, page B gives a vote worth 0.125 to page A and a vote worth 0.125 to page C. Only one third of D's PageRank is counted for A's PageRank which is approximately 0.081. In this case, the PageRank of A can be calculated as:

$$PR(A) = \frac{PR(B)}{2} + \frac{PR(C)}{1} + \frac{PR(D)}{3}.$$

In other words, the value of the vote from a page is equal to its own PageRank score divided by the number of its outbound links. Let L(X) be the number of page X's outbound links. PR (A) can be expressed more generally by

$$PR(A) = \frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)}.$$

Google recalculates PageRank scores each time it crawls the Web and rebuilds its index. As the number of documents in Google's collection increases, the initial approximation

of PageRank for all documents decreases. The formula uses a model of a random surfer who gets bored after several clicks and switches to a random page. The probabilities, at any step, that the person will continue is a damping factor d. Various studies have tested different damping factors, but it is generally assumed that the damping factor will be set around 0.85 [9]. The PageRank value of a page reflects the chance that the random surfer will land on that page by clicking on a link. It can be understood as a Markov chain in which the states are pages, and the transitions are all equally probable and are the links between pages. If a page has no links to other pages, it becomes a sink and therefore terminates the random surfing process. However, the solution is quite simple. If the random surfer arrives at a sink page, it picks another URL at random and continues to surf again.

When calculating PageRank, pages with no outbound links are assumed to link out to all other pages in the collection. Their PageRank scores are therefore divided evenly among all other pages. But to be fair with pages that are not sinks, these random transitions are added to all nodes in the Web, with a residual probability of usually d = 0.85 [9], estimated from the frequency that an average surfer uses his or her browser's bookmark feature. The equation is as follows:

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

where $p_1, p_2, \ldots, p_N$ are the pages under consideration, $M(p_i)$ is the set of pages that link to p_i, $L(p_j)$ is the number of outbound links on page p_j, and N is the total number of pages.

The PageRank values are the entries of the dominant eigenvector of the modified adjacency matrix. This makes PageRank a particularly elegant metric: the eigenvector is

$$\mathbf{R} = \begin{bmatrix} PR(p_1) \\ PR(p_2) \\ \vdots \\ PR(p_N) \end{bmatrix}$$

where **R** is the solution of the equation

$$\mathbf{R} = \begin{bmatrix} (1-d)/N \\ (1-d)/N \\ \vdots \\ (1-d)/N \end{bmatrix} + d \begin{bmatrix} \ell(p_1,p_1) & \ell(p_1,p_2) & \cdots & \ell(p_1,p_N) \\ \ell(p_2,p_1) & \ddots & & \\ \vdots & & \ell(p_i,p_j) & \\ \ell(p_N,p_1) & & & \ell(p_N,p_N) \end{bmatrix} \mathbf{R}$$

where the adjacency function $\ell(p_i, p_j)$ is 0 if page p_j does not link to p_i, and normalized such that, for each j

$$\sum_{i=1}^{N} \ell(p_i, p_j) = 1,$$

i.e. the elements of each column sum up to 1.

This is a variant of the eigenvector centrality measure commonly used in network analysis. The values of the PageRank eigenvector are fast to approximate with only a few iterations and in practice it gives good results. As a result of Markov theory, it can be shown that the PageRank of a page is the probability of being at that page after lots of clicks. This happens to be equal to t^{-1} where t is the expectation of the number of clicks or random jumps required to get from the page back to itself. The main disadvantage is that it favors older pages, because a new page, even with high degree of relevant, will not have many links unless it is part of an existing site.

Drawbacks:

Google has been criticized for placing long term cookies on users' machines to store their preferences, which enables them to track a user's search terms over time. Also some Google skew search results might occur because in order to influence their web sites, some web sites use both on-page factors like body copy, title tags, H1 heading tags and off-page factor like anchor text. The general on-page idea is to affect Google's relevance algorithm by putting the keywords being targeted in various places, in particular the title tag and the body copy because the higher up in the page, the better its keyword prominence and thus the ranking. One off-page technique that they normally used is Google bombing in which websites link to another site using a particular word in the anchor text. Anchor text is the visible text in a hyperlink and is weighted highly in search engine algorithms because the linked text is usually relevant to the landing page. The objective is trying to provide highly relevant search results in order to give the site a high ranking when the word is searched for.

2.2 LYCOS

On March 2, 2005, Lycos [2] announced that they have selected Ask [5] Jeeves' algorithmic which itself use Teoma [10] search technology to do the power search. Based on social networking theory, Teoma technology takes a unique approach to relevancy ranking. Like some search engines, Teoma utilizes a form of link popularity to assist in page ranking, but it provides better results because it goes beyond traditional page ranking methods to determine authority and relevancy. To determine the authority or quality of a site's content, Teoma uses Subject-Specific Popularity. It ranks a site based

on the number of same-subject pages that reference it, not just general popularity. With this Subject-Specific Popularity technology, Teoma is the first search engine to break the Web down into topic-based "communities" of sites, and to give credibility to those sites respected as authorities on a particular subject.

Subject-Specific Popularity analyzes the relationship of sites within a community, ranking a site based on the number of same-subject pages that reference it, among other things. In other words, Teoma determines the best answer for a search by asking experts within a specific subject community about who they believe is the best resource for that subject. By assessing the opinions of a site's peers, Teoma establishes authority for the search result. Finally, by dividing the Web into local subject communities, Teoma [10] is able to find and identify expert resources about that subject.

As we can see with Ask [5] / Lycos [2] search technology, it's not just about who's biggest: it's about who's the best. Their algorithm goes beyond mere link popularity (which ranks pages based on the sheer volume of links pointing to a particular page) to determine popularity among pages considered to be experts on the topic of your search. This is known as Subject-Specific popularity. Identifying topics (also known as "clusters"), the experts on those topics and the popularity of millions of pages amongst those experts , at the exact moment your search query is conducted requires many additional calculations that some other search engines do not perform.

Drawbacks:

Ask is the fourth largest search property in the English language market. So many webmasters try to skew the Ask search results by altering the following factors which are considered as a high ranking for Ask:

- Keywords used within the Title tag
- Keywords used within the Description metatag
- Page content, position and emphasis of keywords
- The number and type of inbound hyperlinks from other relevant and quality websites.

2.3 HOTBOT

Hotbot [3] now offers either Msn or Ask for user to do their search. Since Msn [6] is one of the three major search engines these days, our search engine uses Msn [6] as default. Msn search engine index now includes more than 5 billion documents, 400 million images and 3 million instant answers.

In June, 2005, Microsoft revealed that they have introduced a new ranking algorithm that employs a neural network. Artificial neural networks are models inspired by the study of biological neural networks, and consist of a collection of interconnected units that perform a collective task. The units are modeled using mathematical functions, which themselves depend on the other functions in ways defined by the structure of the network. One of the very desirable features of neural nets is their ability to learn. This is undoubtedly the reason that Msn have chosen to make use of this type of model. The goal of these search engines is to provide the most relevant search results possible for a given search term. The neural net model produces an improved set of results over time, as it learns from past searches. So any search term will improved over a period of a few weeks because the Msn [6] search engine "learned" to produce more relevant results.

Drawbacks:

As with other search engines, all webmasters try to alter their sites to closely follow Msn ranking algorithm in order to boost up their ranking in Msn. As a result, it skews the Msn search results. All the following facts affect Msn ranking:

- The number of links directed to that site.
- The content of the site. This is where much of the ranking determination is made. Sites with great text and clear internal link-paths are ranked very well.
- Strong, keyword enriched titles and body texts continue to provide strong placements.
- Size matters with Msn [6] as larger sites with long-term content appear to be doing very well under more generic keyword searches. Content rich news and information sites and large corporate sites will be able to leverage their size and content-scope into high placements.

2.4 ALTAVISTA

In February 2003, AltaVista [4] was bought by Overture. By March 2004, Overture itself was taken over by Yahoo [7]. In Aug. 2004, shortly after Yahoo acquisition, the AltaVista site started using the Yahoo! Search database. But on 17th February 2004, Yahoo already dropped Google [1] results, and instead showed search results using Inktomi [11] algorithm. Inktomi doesn't provide a public search engine in a way that search engines like Google do. Instead, it licenses its search technology and database to other portals and sites on the Internet. While some of them display results which are taken directly from Inktomi's database, others display results from Inktomi after displaying results from another source. Inktomi is built of two databases WebMap and Web Search 9.

1) **WebMap** - This is not a searchable database. Inktomi claims that it contains three billion documents .This database is used to analyze the characteristics of each page and the link structure that connects the pages said otherwise they are using it to distill link popularity.

2) **Web Search 9** - From WebMap database Inktomi chooses the highest quality documents to build Web Search 9 database, which contains the former databases: Eurocluster, the Asia Pacific cluster, the Best Of Web cluster and GEN3.

Although Inktomi [11] / Yahoo [7] search algorithm doesn't differ too much from that of Google's, it is not exactly a clone of Google's algorithm. The Yahoo Search index captures the full text of web pages, up to a 500K limit. This is greater than the 101K maximum indexed by Google. A broad range of file types, including HTML, PDF, and Microsoft Office documents is also included in the mix. Click Popularity is part of Yahoo's Algorithm and Google doesn't put much on that to determine one website ranking. So the more visitors click on your website, the more you'll get close to the Top Yahoo Ranking. Also the more site-wide linking will rank your site much higher in Yahoo but one link per domain will rank well on Google search engine.

Drawbacks:

Similar to other search engines, all webmasters try to find what kind of elements is Yahoo's search algorithm built of? For long, it's been claimed that including keywords in the title, description and/or URL that submit to Yahoo will boosts their search ranking for

those words. After Yahoo introduced click-through tracking on its pages, click popularity has also been said to be an important factor. So all webmasters try to skew the Yahoo search results by alter those information within their site.

2.5 META SERACH ENGINE RATIONALE

After going through the features of all four search engines, let's talk about Meta Search Engine (MSE) [15]. Since no single search engine can find all relevant pages, users often need to switch from one search engine to another to locate the desired information. Also the availability of an abundance of search engines often makes users confused about which engine to select and under what conditions. In addition, each search engine has its own unique user interface and features and needs different search strategies. There are many articles on the web that teach users about the features of search engines and how to take advantage of each one. General search engine today are also harmed by the noise of blog cross linking, link bombing or Google bombing, and commercial efforts to skew PageRank scores. These facts, together with the limitations of each search engine discussed in the previous section, have led to the introduction of meta search engines.
MSEs Come in four flavors: "Real" MSEs, "Pseudo" MSEs Type I, "Pseudo" MSEs Type II and Search Utilities

1) **"Real" MSEs -** These "Real" MSEs simultaneously search the major search engines, aggregate the results, eliminate the duplicates and return the most relevant matches, according to the engine's algorithm. Two of the example sites in this class are www.vivisimo.com, www.infonetware.com

2) **"Pseudo" MSEs Type I** - The type I "Pseudo" MSE sends the query to the search engines, and then presents the results grouped by search engine in one long, easy to read scrollable list. But based on how many SEs user selects, the waiting time can be very long. Some people might find these MSEs useful. Two of the example sites in this class are www.planetsearch.com, www.searchwiz.com

3) **"Pseudo" MSEs Type II** - You type your query one time and then select the search engines. One browser window will open for each SE selected. Two of the example sites in this class are www.searchbridge.com, www.theinfo.com

4) **Search Utilities (Desktop Search Applications)** - These are downloadable meta search tools that search multiple search engines. Results are collated and ranked for relevancy with redundancies removed. They are not free but most of them have a free trial version available. The two most popular in this class are: www.orbiscope.net, www.searchenginecommando.com

The pros for the first three MSEs are free and can run in different platforms (both window and Linux) but are limit to the uptime of the meta search engine websites. The pro for the last one is that it can search anytime as long as the user PC is up but most of them are limit to the window environment. According to one of the University of Albany research [12], the pros and cons for all the meta search engines are stated as follows:

i) Pros - MSE save searchers a considerable amount of time by sparing them the trouble of running a query in each search engine. In addition, the search results are extremely relevant most of the time.

ii) Cons - Most of the MSE uses are limited primarily to simple queries with little or no field searching. These engines return a limited number of results that do not represent the totality of results from any source engines.

After going through all these pros and cons, we proceeded to design a desktop meta search engine application which runs at the user machine to simultaneously search four other search engines like Google [1], Lycos[2] Hotbot[3], and AltaVista[4]. Figure 2 illustrates the architecture of our search engine. By implementing the meta search engine in the user machine, we eliminate the site downtime problem found in some other MSEs. Also, our meta search engine addresses some of the cons like limited queries stated by the University of Albany research [12]. To do that, our implementation will allow user to do Boolean search . In addition, our implementation allows user a) to choose between window and Linux platforms (which is free and getting more popular); b) to remove foreign title results; c) to set the engine time out to shorten the search waiting time in case one of the basic search engines is down. Final cons stated by the Albany research [12] are due to the nature of meta search which limits the returning results. But we use this limitation as one of our implementation's major features, which is to sort the result titles alphabetically. This unique feature, which no other meta search websites currently have, helps people retrieve the same results at the second time much faster . It is because sometimes user might only remember the link title after the first

search but forget to bookmark it. Thus, sorting results alphabetically allows the user locate the results much faster the second time around. But this sorting feature only works if the number of search results is limited.

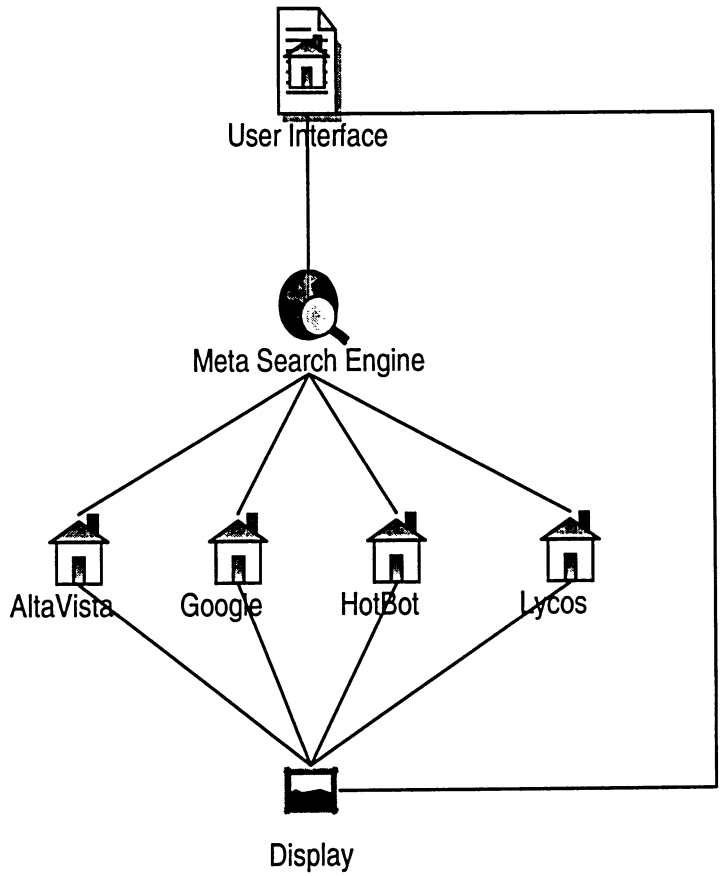

Fig 2. Architecture of our meta search engine

In the next section, we are going to talk about how we use java to implement all the different features for the program.

3.0 OUR META SEARCH ENGINE IMPLEMENTATION FRAMEWORK

As a proof of concept, we build a prototype that implements our proposed meta search engine scheme. The goal is to test the approach and draw some initial conclusions. Specifically, we wanted to see if the system can reliably produce the expected search results depending on the chosen search engines.

We create a file called isearch.jar that includes all files in Table 2 and the gnu-regexp-1.1.4.jar library file. Users can run the program by using java –jar isearch.jar in either window or Linux platform. But we wrote a bat file in window, so users can just click the bat file to start the program.

Once the program starts, users can choose the File option to exit or the Tools option to configurate in the menu bar. If users click the Configuration option, a configuration dialog box will start and provides the following options:

1) Users can choose the number of results per search engine by typing in any positive number, or a drop down box will also appeared to allow users to choose the number of results ranging from 10, 20, 40, 100, or 200 results for their convenience.
2) Users can choose any search engine combination, and by default it is Google and Hotbot.
3) There are two more options for users, one is to remove "foreign" titles and the other is to remove garbage titles.

4) Timeout option is used to allow users to type in any positive number or choose from the drop down choice box of 0, 5, 10, 20, and 60. (Zero means no timeout and waits until finished). Once a timeout option is defined, it affects all active engines and after specified number of second passes, it stops all search threads and forces search engine to return any results they already have.

5) The last option is to let user to choose a different Brower to display the search results which is a requirement because platforms such as Linux doesn't have Internet Explorer.

After users make all the changes, they can either press ok or cancel to go back to the main window .They can then enter what they want to search and press the "Search" button to start searching. In order to make the program work like Internet Explorer or other Browsers of having "URL" field drop down menu which including all recent searches, we set up a HistoryComboBox ,an extension to JComboBox class which remembers previously entered values and allows user to select them later from the drop-down menu. Finally, user can choose the two sort options which are sort by relevance or alphabetically inside the main window too.

We are going to talk about the three components that Dreilinger and Howe [16] suggest for a successful meta search engine. As mentioned in the previous section, they are Interface Agents, Dispatch Mechanism and Display Mechanism .And Table 2 outlines all the programs we wrote for those three requirements:

Components	Folder	Java Application
Interface Agents	kwokchan\isearch\engine	AltavistaAdaptor.java GoogleAdaptor.java HotbotAdaptor.java LycosAdaptor.java
Dispatch Mechanism	kwokchan\isearch\gui	Config.java ConigWindow.java EngineBox.java HistoryComboBox.java Isearch.java IsearchConstants.java MainFrame.java ProgressWindow.java
Display Mechanism	kwokchan\isearch	EngineAdaptor.java EngineRegistry.java EngineRequest.java EngineSet.java EngineThread.java Hit.java HTMLTools.java MutableBoolean.java Searcher.java

Table 2. Program written for our meta search engine

3.1 INTERFACE AGENTS

Inside the isearch.jar file, we included the java regular expression package which can be downloaded from http://www.crocodile.org/~sts/Rex/. In order to find out the URL within each search results, we have to find out what URL the search engine uses to produce results, then use that URL to construct HTTP request. We then run the search manually to see how the results are formatted in the HTML code returned by that search engine. After this step, we use JAVA regexp class [13] to find out the search hit hyperlinks in HTML code and distinguish them from everything else (advertisements, etc) .These patterns are different for different engines. Since search engines can only produce fixed number of results per page, we have to look for 'Next Page' hyperlink and follow it to get the next page results before merging.

To enable Windows Look & Feel for the whole Java application (Windows only) and also make it to use Windows colors, fonts, styles of buttons like other regular Windows application does, we use Swing UI manager setLookAndFeel method. This method is implemented based on the following prototype:

UIManager.setLookAndFeel ("com.sun.java.swing.plaf.windows.WindowsLookAndFeel")

Because of that, users have at least JDK 1.2 installed in their systems which support swing in order to make the program run.

For our program, there are four major Interface agents. They are AltavistaAdaptor.java, GoogleAdaptor.java, HotbotAdaptor.java and LycosAdaptor.java

1) **AltavistaAdaptor class -** To submit a search request to Altavista web site (with a word Metal, for example), we craft the following URL:

http://www.altavista.com/web/results?q=metal&nbq=40&kgs=0&kls=0

This is the same URL users gets when they manually search the word "Metal" on www.altavista.com [4] and press the 'Search' button. When users send such request to the search engine web server, one of the HTML sources of the page returned by a search word "Metal" on Altavista.com among other things will have some expressions like this:

```
<a class='res'
href='http://av.rds.yahoo.com/_ylt=A0Je5XOFTaxA5TYA3mBrCqMX;_ylu=X3oDMTB
vdmM3bGlxBHBndANhdl93ZWJfcmVzdWx0BHNlYwNzcg--
/SIG=10tp6c9m4/**http%3a//www.encycmet.com/'>ENCYCLOPEDIA
<b>METAL</b> - For fans, by fans.</a>
```

We use JAVA regexp [13] to analyze the above response and look for any text within the response which satisfies the following regular expression pattern that we found manually:

```
<a\\s+class='res'\\s+href\\s*=\\s*'[^']+\\*\\*(http[^']+)'[^>]*?>(.*?)</a>
```

Our regular expression pattern matches the above HTML expression result as they all begin with and their structure conforms to the rest of the regular expression pattern above. As a result, this is a search hit hyperlink returned by AltaVista.

2) **GoogleAdaptor class -** To submit a search request to Google web site, we craft the following URL:

http://www.google.com/search?q=

This is the same URL users get when they press the "Search "button in Google. When they send such a request to the search engine web server, the web server returns HTML document with all search results. One of the HTML sources of the page returned by a search word "test" on Google.com among other things will have some expressions like this:

NerdTests.com Fun Tests - Nerd Quiz

Again, we use JAVA regexp [13] to analyze this response and look for any text within that response which match the following regular expression pattern that we found manually:

<a\\s+href\\s*=\"(.*?)\"\\s+class=l\\s*>(.*?)

Our regular expression pattern matches the above HTML expression result as they all begin with <a href ...class=1...> and their structure conforms to the rest of the regular expression pattern above. As a result, this is a search hit hyperlink returned by Google.

3) **HotbotAdaptor class** - To submit a search request to Hotbot web site, we craft the following URL:

http://www.hotbot.com/default.asp?query=

This is the same URL users get when they manually press the 'Search' button at Hotbot. When users send such a request to the search engine web server, one of the HTML sources of the page returned by a search, among other things will have some expressions like this:

Test.com Web Based Testing Software

We use JAVA regexp [13] to analyze this response and look for any text within the response which satisfies the following regular expression pattern that we found manually:

<a[^>]+href\\s*=\"([^\"]+)\"[^>]*>([^\n\r]*?)

Our regular expression pattern matches the above HTML expression result as they all begin with <a href....> and their structure conforms to the rest of the regular expression pattern above. As a result, this is a search hit hyperlink that returned by HotBot.

4) **LycosAdaptor class** - To submit a search request to Lycos web site, we craft the following URL:

http://search.lycos.com/default.asp?loc=searchbox&tab=web&query=

This is the same URL users get when they manually press the 'Search' button at Lycos. When users send such a request to the search engine web server, the web server returns HTML document with all the search results. One of the HTML sources of the page returned by a search, among other things will have some expressions like this:

<a href="http://sidesearch.lycos.com/?page=0&query=test&hurl=http%3A%2F%2
Ftmsyn.wc.ask.com%2Fr%3Ft%3Dan%26s%3Drz%26uid%3D24a7ba6e74a7ba6e7%26s
id%3D34a7ba6e74a7ba6e7%26o%3D0%26qid%3DC037FCBEC1D6FF96E762E696E5
6517FB%26io%3D9%26sv%3D0a300589%26ask%3Dtest%26uip%3D4a7ba6e7%26en
%3Dte%26eo%3D7%26pt%3DAndrew%2B%2526quot%253BTest%2526quot%253B%
2BMartin%2B%2BWikipedia%252C%2Bthe%2Bfree%2Bencyclopedia%26ac%3D20%
26qs%3D121%26pg%3D1%26u%3Dhttp%3A%2F%2Fen.wikipedia.org%2Fwiki%2FA
ndrew_%2522Test%2522_Martin"onmouseouver="window.status='http://en.wikipedia.or
g/wiki/Andrew_%22Test%22_Martin'; return true;" onmouseout="window.status=";
return true;"target="_search"onClick="location='http://tmsyn.wc.ask.com/r?t=an&s=
rz&uid=24a7ba6e74a7ba6e7&sid=34a7ba6e74a7ba6e7&o=0&qid=C037FCBEC1D6FF9
6E762E696E56517FB&io=9&sv=0a300589&ask=test&uip=4a7ba6e7&en=te&eo=7&pt
=Andrew+%26quot%3BTest%26quot%3B+Martin++Wikipedia%2C+the+free+encyclop
edia&ac=20&qs=121&pg=1&u=http://en.wikipedia.org/wiki/Andrew_%22Test%22_Ma
rtin';" class="large">Andrew "Test" Martin - Wikipedia, the free
encyclopedia

Again, we use JAVA regexp [13] to analyze this response and look for any text within that response which match the following regular expression pattern that we found manually:

<a\\s+href\\s*=\"([^\"]+)\\\s*[^>]*>([^\r\n]+)

Our regular expression pattern matches the above HTML expression result as they all begin with <a href....> and their structure conforms to the rest of the regular expression pattern above. As a result, this is a search hit hyperlink that returned by Lycos.

3.2 DISPATCH MECHANISM

3.2.1 ConfigWindow class

Configuration dialog box GUI is implemented by one of our program classes called ConfigWindow, which use the Jdialog class (a subclass of java.awt.Dialog) to provide the swing dialog box. And the following are some of the features we want to mention inside our program ConfigWindow class:

1) To get the search engine choice, we use the GridBagConstraints class which defines the layout for the four search engines choices .Since we give user one choice per row, we can capture the engine id which corresponds to the second entry in GridBagConstraints.

2) To make the default Browser option GUI physically works on Windows, we create a temporary shortcut file, having ".url" extension, and launch it using the following command:"rundll32 url.dll,FileProtocolHandler file.url" and this automatically invokes default Browser.

3) For the rest of this configuration dialog box GUI layout, we combine GridBagConstraints and Insets class to determine the amount of space between different panels edges .As a result we can specify the border for different GUI components.

3.2.2 Graphical User Interface (GUI)

There are three GUI classes of our project, which are HistoryComboBox, ProgressWindow and MainFrame classes.

1) **HistoryComboBox class** - In order to have the search box drop down menu to contain user search history, we create HistoryComboBox class to keep user past 10 search entries. If more than 10 entries capture inside the history box, it removes the oldest entry which is at the bottom of the drop down list. The following [Fig 3] is the screenshot capture for the drop down box

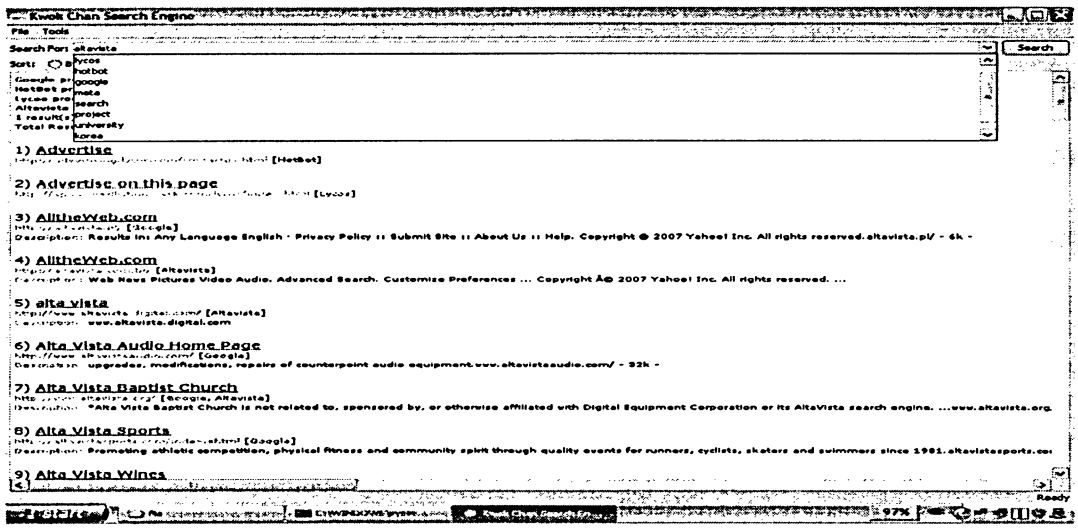

Fig 3. Drop down history box

2) **ProgressWindow class** - Since we want to use colored text on the main frame to display search results, it causes some delay on showing all the search results due to the way JDK hyper-text component works. So, in order to let users know the search engine is still doing the searching, we create the ProgressWindow class, and after user clicks the search button, the progress windows popup, and once the first page of results shown, a "loading" word displays at the bottom right hand corner. After collecting all the results,

the word changes to "Ready". Now user can click on any of the result titles, and the search engine will then open a separate Browser and display the web page corresponding to that search title. Because of the time-consuming search task is performed at the background, the purpose of showing the progress window will act like a task controller .Also, in order to allow user to cancel the search after it starts, a cancel button is implemented inside the popup window. And if users click at the right time, it cancels one of the search engines and display "cancel" as a search result for that search engine. The following [Fig 4] is the screenshot capture for the progress pop up window.

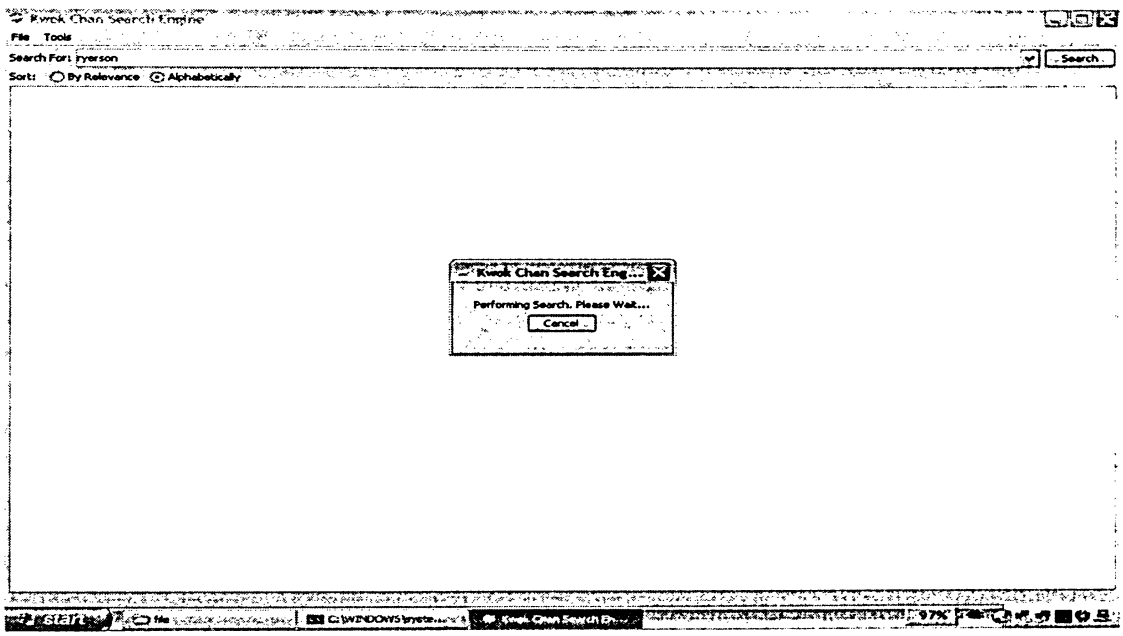

Fig 4. Progress pop up window

3) **MainFrame class** - It is the main display that shows all the search results. We implement some options within this class to allow user to start searching by either click the 'search' button or use the hot key Alt-S, and it also allows user to use hotkeys for all the options within the menu bar like Alt-F (File), Alt-T (Tools).

In order to let user know right away the ranking for all the search results, we implement this class to display the number corresponding to the title ranking plus ") "before each search title. For example, if the result title is the first one, it displays a"1) "and then follows by the search title. Also if there have some problems during the search like the user PC is not on internet. Then it will display "Nothing found" instead.

After all search results are displayed inside the main windows, users can move the mouse on top of any search result titles and the link which corresponds to that search title will display at the bottom left hand corner. They can also make the link disappear by moving the mouse away from the title.

Also within this class, a subclass catch users errors if they choose the custom Browser option in the ConfigWindow but forget to input the command line location .As a result , our search engine will pop up an error window and display " Browser command line is not specified. Open configuration and specify browser command." messages. And it also catches errors when users input a wrong Browser location and because of that, it display "Couldn't launch browser" messages. Besides all that, we also use the getMessage method which presents in all JAVA exception to display a more detailed error messages about what actually happened.

3.3 DISPLAY MECHANISM

The following are two of the classes that are used to do the general search which are the HTMLTools and Searcher classes

3.3.1 HTMLTools

1) To remove HTML comments from the search results string, we implemented subclass within the HTMLTools class to extract what is in between "<!--.*?--> "

2) Within the HTMLTools class, a subclass extracts <BODY> section from the HTML search result and if no <BODY> tag is found in the text, or no closing </BODY> tag is found, the whole original text returned

3) Since there is no 100% reliable way to determine if the search result title is what we call "foreign" in our program (which might be a real foreign character or just some non-English characters). And we cannot just reject a title because it contains non-Latin characters since it's quite common to spell the word "resume" with accented 'e' and most of us still consider it is an English word. Therefore, in order to be more practical, we choose a more common sense algorithm to detect "foreign" title, which works for most cases. And because of that, even user sometimes checked the remove "foreign" title option; our search still shows some non-English titles results. However, during our testing throughout the design of this project, our search still provides very good results.
To define whether the search results are what we call "foreign" title, we implemented the following three definitions to determine whether the result is "foreign" title. If titles match any one of the definition, we consider them as "foreign" title for our search engine.

i) Definition 1:

The following consider search results as foreign if code greater than 255 , because it mean it is a non-ASCII Unicode and, therefore, we can treat it as "foreign " .For example like Chinese text ,it contains explicit foreign Unicode sequences of"中"

```
while( (match=unicodeRE.getMatch(s, offset)) != null) {
int code = atoi(match.toString(1));
if (code > 255) {
        return true;
```

ii) Definition 2:

We use the following common sense algorithm to define "foreign" title for those non-English titles. a) A title has more than 5 extended ASCII characters. b) The number of such characters is greater than the number of Latin characters. c) There are more than half of those characters after removing all the spaces within the title itself

If "number of (code > 127)" > 5 then
 If "number of (code > 127)" > "number of English letter characters for the title" then
  If "number of (code > 127)" > "50% of the total character for the title but excluding space in-between words" then
   The title is "foreign" for our program

iii) Definition 3:

We use another common sense algorithm to define the search tile as "foreign" for those titles with a lot of "?" and "." .The following consider result search title as "foreign" for our program if it consists of at least 75% (instead of just 50% as in Definition 2 because those two characters are commonly used at the end of each English sentence) of '?' and '.' out of at least 12 characters but excluding all the space in between words. "?" symbols are when the Browser does not know what to do with an encoded character which is outside of the range of the character set

```
if (totalCount >= 12 && punctCount > (totalCount / 4 * 3))
    return true;
```

4) During the testing of our program, we have some "Untitled" and "No titles" search results, so we add the remove "garbage" titles option, which is completely independent of the first one. This option is useful to users if this is their second time search since most users in the internet will not go through the "No title" hyperlink. By checking this option, users will be able to identify the required title within a much shorter timeframe. However, if the users found a "No title" search result useful, then there is no need for them to use this "garbage" option. As a result, we create the "garbage" option as a second independent option.

To implement the "Aggressively remove garbage titles" option which works independently from the above "foreign" filter, we set up the following definition to determine whether the search results are garbage titles or not:

i) Definition 1:

The following consider search result title as "garbage" title if result title is called 'Untitled', 'Untitled Document', 'No title', or 'Cached' with regardless of letter case and spaces

```
  private static final RE untitledRE = new
UncheckedRE("\\W*untitled(\\s+document)?\\W*", RE.REG_ICASE);
  private static final RE noTitleRE = new UncheckedRE("\\W*no\\s*title\\W*",
RE.REG_ICASE);
  private static final RE cachedRE = new UncheckedRE("\\W*cached\\W*",
RE.REG_ICASE);
```

ii) Definition 2:

The following consider search result title as "garbage" title if the title contains either a) 4 or more consecutive dots or questions mark or b) It starts with a digit, dot or question mark

```
private static final String badFirstChar = "01234567890?.";
  public static   Boolean isGarbage(String html) {
    if (isEmpty(html)) return true;
    if (html.indexOf(fourDots) != -1) return true;
    if (html.indexOf(fourQuestions) != -1) return true;
```

iii) Definition 3:

The following define result title as "garbage" title if after removing spaces; the title contains at least "title length' - 2 dots and question marks (- 2 because a question mark or dot and one space at the end is a normal English sentence format)

```
int totalCount = 0, punctCount = 0;
    for(int i=0,n=html.length(); i<n; i++) {
      c = html.charAt(i);
      if (spaces.indexOf(c) != -1) continue;
      totalCount++;
      if (qdot.indexOf(c) != -1) punctCount++;
    }
    if (punctCount >= totalCount-2)
      return true;
```

3.3.2 Searcher

The searcher is used to implement all the above HTML Tools foreign and garbage titles definitions. The following display message in the main page like "# result(s) were removed as foreign or "garbage"" or "the total number of results and the total time of searching":

```
    if (foreignRemoved > 0) {
      stats.append("<B>"+foreignRemoved+"</B> result(s) were removed as
foreign<BR>\n");
    }
    if (garbageRemoved > 0) {
      stats.append("<B>"+garbageRemoved+"</B> result(s) were removed as
\"garbage\"<BR>\n");
    }
    if (results.size() > 0) {
      long totalTime = System.currentTimeMillis() - t1;
      stats.append("Total Results: <B>"+results.size()+"</B> ("+totalTime+"
ms)<BR>\n");
    }
```

4.0 SIMULATIONS

After talking about all the java implementation for our program in the last section, we are going to talk about our tests results for the program in this section.We conducted various simulations to assess the proposed meta search engine model and the proposed algorithm.

4.1 APPROACH AND SETUP

We use JAVA to do this project because it can run on any platform without much modification. But user has to has JDK [14] or JRE [14] installed on that platform because the user's PC has to support swing which we used to give those nice HTML text component. Search can be start from command line like this: "java –jar isearch.jar"

4.1.1 Configuration

1) Users can choose any combination of the following four search engines to do their search: Google [1], Lycos [2], Hotbot [3], or AltaVista [4]. According to our test results, Lycos [2] is the slowest search site of the four engines most of the time, and the search times displayed beside the number of results prove that. Google [1] produces more hit and AltaVista [4] is the closest second on most of our testing.

2) User can set the number of search results per engine and this can be set up to 200 results per engine. However if supports more than 200 results, it create more problems for the sort by alphabetically option because more "less relevant results" will appeared at the top of the search results if those titles start with an "A''.Using a maximum of up to 200 results per engine is enough because almost 90% of web searchers are interested in first few search result pages anyway.

3) User can set the stop time for the search and this can be set from 60 sec to no time out

4) User can choose different Web Browsers. Initially, default Windows Browser is used (on Windows only). %1 is placeholder for URL that OS command used to start Browser. For example, if users want to specify firefox as their Browser, they can input "c:\Program Files\Mozilla firefox\firefox %1" for Windows or "/firefox/./firefox %1" for Linux in the "specify command line:" dialog box.

5) User has the Option to remove "Foreign" titles from the search results (Good for normal search and use the sort by relevance option).

6) User has another option to remove "garbage" titles (Good for sort by aphetically search option). It works independently of 'Foreign' filter.

All the above options are shown in [Fig 5] below.

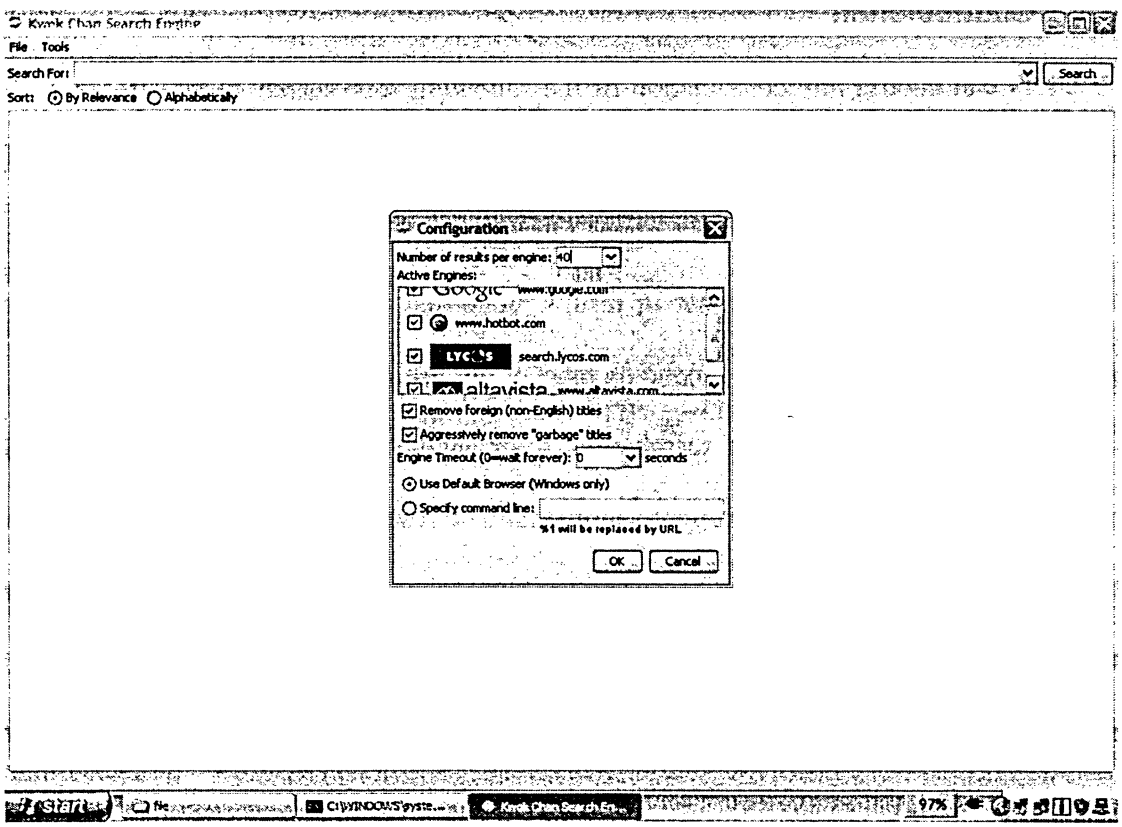

Fig 5. Configuration pop up window

For all display results, user can click the blue link within the window below to get to the corresponding site. The following two screenshots are the search results for the word "Korea" without or with the "garbage" option are checked. [Fig 6 and 7]

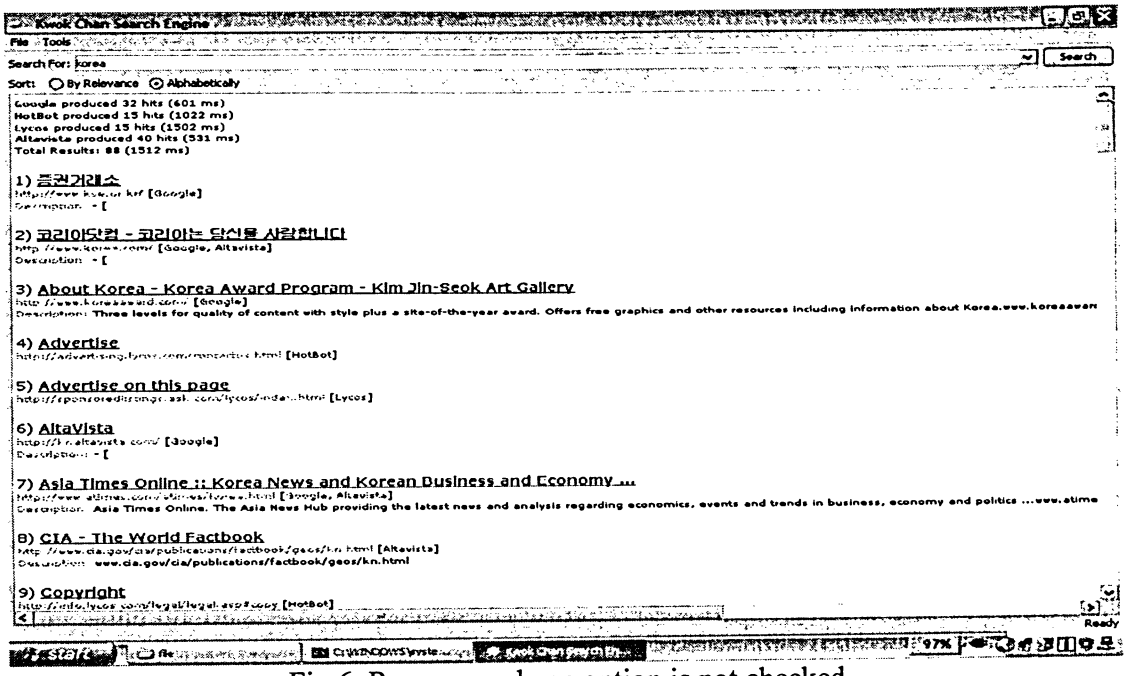

Fig 6. Remove garbage option is not checked

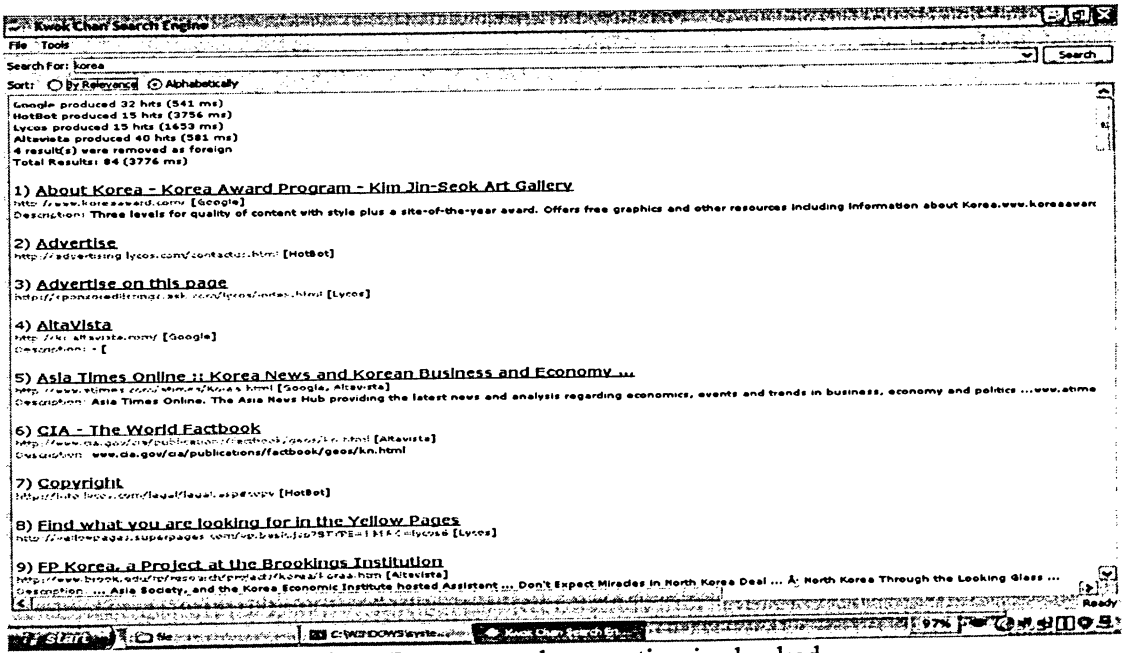

Fig 7. Remove garbage option is checked

7) Our search engine remembers previously entered search values to allow user to select them later from the drop-down menu, just like Internet Explorer and other Browsers do for the "URL" field. However, if more than 10 entries capture inside the history box, the oldest entry which is at the bottom of the drop down list is removed.

4.2 RESULTS

If any one of the engines picked up a description field from the search, our search engine will display the same description. For example, some Google [1] search results have a "Description:" field, and our program can display the same "Description:" field. Otherwise, our program takes whatever comes between result hyperlink and meta-information. To show that, we use Google and Hotbot as our example:

1) **Google -** We use JAVA regexp [13] to analyze the returns HTML document and look for any text within that response which matches the following regular expression pattern that we found manually in order to get the "Description" from some of the Google search hit hyperlink:

Description :(.*?)

2) **HotBot -** For example, one of the search hit returned by a HotBot search, among other things have some expressions like this:

Test.com Web Based Testing Software Provides extranet privacy to clients making a range of tests and surveys available to their human resources departments. Companies can test prospective and current employees. Information on surveys ...

From the above return HTML expression , we just extract the description which is

"Provides extranet privacy to clients making a range of tests and surveys available to their human resources departments. Companies can test prospective and current employees. Information on surveys ...".

4.2.1 Sort By Relevance

Since all normal search engines return results is already in relevance order, there is no need to tag the search ranking as the most relevant entries often come first. Before merging results, we assign numeric rank to every hit on an individual engine. (Lower rank indicates higher relevance). When our program merges the search results from two or more engines, we take the average of all the individual engine rank, minus the number of engines that produced the hit, so that multi-engine hits receive higher priority (Smaller value mean higher ranking).

This is the option that user should use during the first time search to get the result that closely match their search item because the most relevance result will be on top of our search results. Figure 8 shows the screenshot for the search result using "Ryerson University"

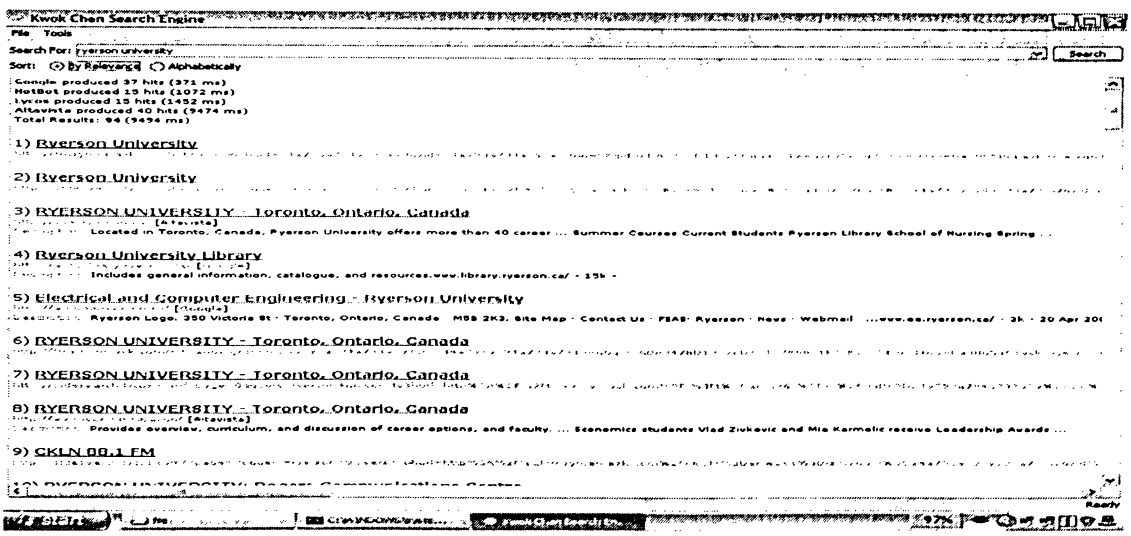

Fig 8. Sort by relevance option is chosen

4.2.2 Sort By Alphabetically

This function is done by sorting the first character of the search title result. However, if we use this option, the following title still show up at the top of the search result list even we check the remove "foreign" title option

"???? ????. on-line Ping Pong. flash game"

It is because the ASCII code represent the "?" is lower than all the other Latin English letter and this phrase contains no "foreign" characters and there are more English Latin characters in it than question marks and, therefore, it is not removed with just the remove 'foreign" option . Because of that, we create another option which is the remove "garbage" option, which is a perfect fit to remove a title like this. However this title might still contain useful information for the user during the first time search (we suggest to use the sort by relevancy option to get all those useful results up on top of the list). So this separate option is created for our program. Figure 9 shows the screenshot capture using the search word "Ryerson University"

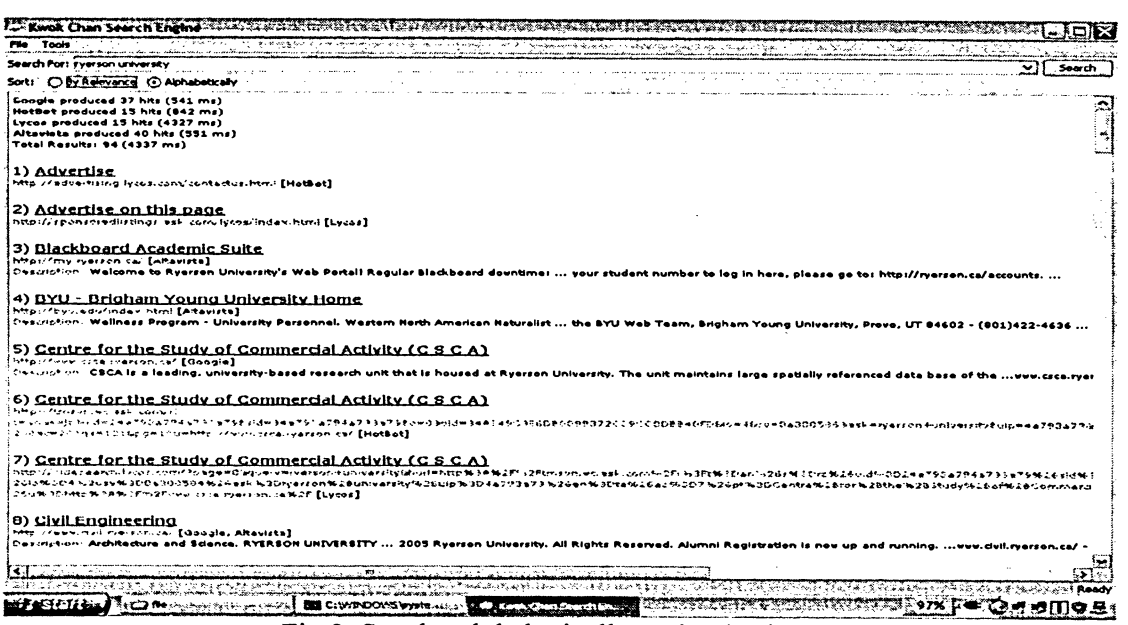

Fig 9. Sort by alphabetically option is chosen

4.3 COMPARSION RESULTS WITH OTHER META SEARCH ENGINES

To be fair, our meta search engine was tested in window platform environment (because other MSE applications can't run in Linux) against eight other MSEs for a month. Table 3 shows the average speed and coverage of 60 different search queries that we used for each test period against each MSE.

MSE Type	Search Engine	Average Speed	Average Coverage	Drawback
"Real" MSE	www.vivisimo.com	3 sec	223 results	Cannot sort result title alphabetically
	www.infonetware.com	Site is down	Site is down	Site is down
"Pseudo" MSE Type I	www.planetsearch.com	7 sec	180 results	Cannot sort result title alphabetically
	www.searchwiz.com	No results	No results	
"Pseudo" MSE Type II	www.searchbridge.com	6 sec	23 results	Displayed different search engines results in separate windows. So it can't minimize the ranking skew effect
	www.theinfo.com	10 sec	Displayed all search results from different engines	
Search Utilities	www.orbiscope.com	2 sec	20 results	Only runs on windows
	www.searchenginecommando.com	15 sec	75 results	
Our Meta Search Engine		5 sec	182 results	

Table 3. Comparison results with other MSEs

From the above search results, our search engine has no site downtime like some of the MSE sites. Plus our search engine can run on both Linux and window platforms (NOT with other MSEs application), and also have the option to sort result titles alphabetically (NOT with other MSE websites). In addition, our speed and coverage are at the top three among the other MSEs. However there are some drawbacks to our program like a) The result URLs are obtained by parsing the returned results pages from underlying engines based on our knowledge of their result page formats. If an engine changes its results display, we have to modify our code correspondingly. b) The query is passed to selected search services simply as it is without any refinement according to each service's query syntax. It is therefore impossible for this prototype to take advantage of the advanced features of each of the underlying search engines.

5.0 CONCLUSION

Our meta search engine has the capability to do a combined search on four different underlying major search engines such as Google, Yahoo, Msn and Ask at the same time. It allows users to take advantage of the different search ranking and sorting algorithm used by each of the four search engines to minimize the skewing ranking effects.

Secondly, since our search engine runs on user's own computer, there will be no site downtime problem like all of the MSE websites. Also, because our users have an option to choose different Browser and our search engine is written in Java, our program has the advantage to work on many different platforms and setups that some other meta search applications cannot do.

Thirdly, our users can have an option to remove foreign title and the capability to shorten the engine search waiting time if one of the basic search engines is down.

Finally, our search engine has even managed to utilize the disadvantages of all meta engines' limited search results to become one of our application features which is to allow users to sort result titles alphabetically from the previous relevancy search result, so that the users can identify the required title within a shorter timeframe.

In conclusion, our meta engine allows multiple engine searches to minimize biased information and improves the quality of the results it generates. However, further research is needed to fully take advantage of the proposed MSE framework. Since its current form would not be able to accommodate any future improvement in the way any of the relevant search engine might change its results display.

6.0 REFERENCES

[1] Google: Http: // www. Google.com/. 2007

[2] Lycos: Http: //www.lycos.com/. 2007

[3] HotBot: http: //www. Hotbot.com/. 2007

[4] AltaVista: http://www.altavista.com/. 2007

[5] Ask: http://www.ask.com/. 2007

[6] Msn: Http: //www.msn.com/. 2007

[7] Yahoo: Http: //www.yahoo.com/. 2007

[8] PageRank (Wikipedia): http://en.wikipedia.org/wiki/PageRank. 2007

[9] Sergey Brin and Lawrence Page , "The anatomy of a large-scale hypertextual Web search engine ". Proceedings of the seventh international conference on World Wide Web 7: 107-117 (Section 2.1.1 Description of PageRank Calculation). 1998

[10] Ask Technical Support: http://sp.ask.com/en/docs/about/webmasters.shtml. 2007

[11] Inktomi Technical Support: http://support.inktomi.com/Search_Engine/Product_Info/FAQ/searchfaq.html. 2007

[12] Albany Research: http://library.albany.edu/internet/meta.html. 2007

[13] Regular Expression: http://java.sun.com/docs/books/tutorial/essential/regex/. 2007

[14] Sun Developer Network: http://java.sun.com/j2se/1.4.2/download.html. 2007

[15] Daniel Bazac, http://www.web-design-in-new-york.com/meta_search_engines.html. 2002.

[16] D. Dreilinger and A. E. Howe, "Experiences with Selecting Search Engines Using Metasearch," *ACM Transactions on Information Systems* , Vol. 15, No. 3, pp. 195 – 222. 1997.

7.0 APPENDIX

We store everything into one jar file which is called isearch.jar. Inside the jar file there have the followings directory:

- **src/** - contains actual java source
- **lib/** - contains free gnu.regexp library in complied form to perform HTML parsing using regular expressions

And our application consists of these packages:

- **kwokchan.isearch** - search engine common functionality
- **kwokchan.isearch.engine** - implementations of search engines
- **kwokchan.isearch.gui** - user interface

The following are all the java class within that isearch.jar file

1) EngineAdaptor.java

```
package kwokchan.isearch;

import java.io.IOException;
import java.util.ArrayList;
import javax.swing.Icon;

public interface EngineAdaptor {

   public String getID();
   public String getName();
   public String getHTMLName();
   public String getEngineURL();
   public Icon getIcon(int iconType);
   public ArrayList parse(String html, EngineRequest request);
   public String[] performRequest(EngineRequest request, MutableBoolean interuptor,
long deadline) throws IOException;

}
```

2) EngineRegistry.java

```
package kwokchan.isearch;

import java.util.HashMap;
import java.util.Set;
import java.util.Vector;

public class EngineRegistry {

   private static HashMap adaptors = new HashMap();
   private static HashMap idIndex = new HashMap();
   private static Vector ids = new Vector();

   public static EngineAdaptor getAdaptorByName(String name) {
      return (EngineAdaptor) adaptors.get(name);
   }


   public static EngineAdaptor getAdaptorByID(String id) {
      return (EngineAdaptor) idIndex.get(id);
   }

   public static String[] getAdaptorNames() {
      Set keySet = adaptors.keySet();
      if (keySet == null) return null;
      else return (String[]) keySet.toArray(new String[keySet.size()]);
   }

   public static void addAdaptor(EngineAdaptor adaptor) {
      adaptors.put(adaptor.getName(), adaptor);
      ids.add(adaptor.getID());
      idIndex.put(adaptor.getID(), adaptor);
   }

    public static String[] getEngineIDs() {
      return (String[]) ids.toArray(new String[ids.size()]);
   }

   public static EngineAdaptor[] getAllAdaptors() {
      EngineAdaptor[] engines = new EngineAdaptor[ids.size()];
      for(int i=0,n=engines.length; i<n; i++) {
         engines[i] = (EngineAdaptor) idIndex.get(ids.get(i));
      }
      return engines;
   }
```

```
    static {
        addAdaptor(new kwokchan.isearch.engine.GoogleAdaptor());
        addAdaptor(new kwokchan.isearch.engine.HotbotAdaptor());
        addAdaptor(new kwokchan.isearch.engine.LycosAdaptor());
        addAdaptor(new kwokchan.isearch.engine.AltavistaAdaptor());
    }

}
```

3) EngineRequest.java

```
package kwokchan.isearch;

public class EngineRequest {
    public static final int OMIT_DESCRIPTIONS = 0x00000001;
    public static final int FOREIGN_FILTER    = 0x00000002;
    public static final int GARBAGE_FILTER    = 0x00000004;
    protected String pattern;
    protected int startIndex, matchCount;
    protected int flags;
    protected int timeout;

    public EngineRequest(String pattern, int startIndex, int matchCount) {
        this.pattern = pattern;
        this.startIndex = startIndex;
        this.matchCount = matchCount;
    }

    public void setFlags(int flags) { this.flags = flags; }
    public boolean isFlagSet(int flag) { return 0 != (this.flags & flag); }
    public void setTimeout(int timeout) { this.timeout = timeout; }
    public int getTimeout() { return timeout; }
    public final String getPattern() { return pattern; }
    public final int getStartIndex() { return startIndex; }
    public final int getMatchCount() { return matchCount; }

}
```

4) EngineSet.java

```
package kwokchan.isearch;
import java.util.ArrayList;
public final class EngineSet  {
    private ArrayList engines;
    public EngineSet(String[] engineIDs) {
        engines = new ArrayList();
```

```
        if (engineIDs != null) {
            for(int i=0,n=engineIDs.length; i<n; i++) {
                EngineAdaptor engine = EngineRegistry.getAdaptorByID(engineIDs[i]);
                if (engine != null)
                    engines.add(engine);
            }
        }
    }

    public EngineAdaptor[] getAllEngines() {
        return (EngineAdaptor[]) engines.toArray(new EngineAdaptor[engines.size()]);
    }

}
```

5) EngineThread.java

```
package kwokchan.isearch;
import java.util.ArrayList;
public class EngineThread implements Runnable {

    private boolean ready;
    private Throwable exception;
    private ArrayList results;
    private EngineAdaptor engine;
    private EngineRequest request;
    private MutableBoolean interuptor;
    private StringBuffer stats;
    private Object synchronizer;

    public EngineThread(EngineAdaptor engine, EngineRequest request, MutableBoolean
interuptor, Object synchronizer) {
        this.engine = engine;
        this.request = request;
        this.interuptor = interuptor;
        this.synchronizer = synchronizer;
        results = new ArrayList();
        stats = new StringBuffer();
    }

    boolean isReady() { return ready; }
    public Throwable getException() { return exception; }
    public ArrayList getResults() { return results; }
    public String getStats() { return stats.toString(); }
    public void run() {
        ready = false;
```

```
EngineAdaptor adaptor = this.engine;
String htmlName = adaptor.getHTMLName();
try {
      long t1 = System.currentTimeMillis();
   ArrayList engineResults = this.results;
   engineResults.clear();
   stats.setLength(0);
    int timeout = request.getTimeout();
long deadline = (timeout > 0) ? (System.currentTimeMillis() + timeout * 1000L) : 0;
   String[] pages = adaptor.performRequest(request, interuptor, deadline);
    boolean missedDeadline = (deadline > 0) &&
       (System.currentTimeMillis() >= deadline);
   if (interuptor.getValue()) throw new InterruptedException();
   String allPages = "";
   if (pages != null) {
      if (pages.length == 1) allPages = HTMLTools.getBody(pages[0]);
      else  {
         for(int i=0,n=pages.length; i<n; i++) {
          allPages = allPages + pages[i];
         }
      }
   }
   if (interuptor.getValue()) throw new InterruptedException();
   engineResults = adaptor.parse(allPages, request);
   this.results = engineResults;
   for(int i=0,n=engineResults.size(); i<n; i++) {
      ((Hit) engineResults.get(i)).setRank(i+1);
   }
   long duration = System.currentTimeMillis() - t1;
   stats.append("<B>"+htmlName+"</B> produced "+engineResults.size()+
     " hits ("+duration+" ms)");
   if (missedDeadline) stats.append(" - <B>timed out</B>");
   stats.append("<BR>\n");
  } catch (Throwable ex) {
    if (interuptor.getValue()) {
      stats.append("<B>"+htmlName+"</B> cancelled.");
      if (!results.isEmpty()) stats.append(" Result set is incomplete.<BR>\n");
    } else {
      stats.append("<B>"+htmlName+"</B> failed: "+ex.getMessage()+"<BR>\n");
      this.exception = ex;
      ex.printStackTrace();
    }
  } finally {
    ready = true;
    synchronized(synchronizer) {
      synchronizer.notify();
    }
```

```
    }
    }
  }
```

6) Hit.java

```
package kwokchan.isearch;
import java.text.MessageFormat;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.Iterator;

public class Hit implements Cloneable, Comparable {
   private String url, title, description;
   private ArrayList engines;
   private int rank;
   private int descriptionWeight;
   public Hit(EngineAdaptor engine, String url, String title, String description) {
      this.url = url;
      this.title = title;
      this.description = description;
      engines = new ArrayList(1);
      if (engine != null) engines.add(engine);
   }
  public String getURL() { return url; }
      void setTitle(String title) {
      this.title = title;
   }
   public String getTitle() { return title; }
   public String getDescription() { return description; }
   public EngineAdaptor[] getEngines() {
      return (EngineAdaptor[]) engines.toArray(new EngineAdaptor[engines.size()]);
   }
   public String getEngineNames() {
      StringBuffer names = new StringBuffer();
      for(int i=0,n=engines.size(); i<n; i++) {
         EngineAdaptor engine = (EngineAdaptor) engines.get(i);
         if (i > 0) names.append(',').append(' ');
         names.append(engine.getHTMLName());
      }
      return names.toString();
   }
      private static MessageFormat hitFormat = new MessageFormat(
      "<A HREF=\"{0}\"><B>{1}</B></A><BR>"+
      "<FONT SIZE=-2><FONT COLOR=gray>{0}</FONT> [{2}]<BR>{3}</FONT>"
   );
```

```
private static MessageFormat descriptionFormat = new MessageFormat(
   "<FONT COLOR=gray>Description:</FONT> {0}<BR>"
);
public void pasteHTML(StringBuffer out) {
   Object[] fields = new Object[4];
   fields[0] = url;
   fields[1] = title;
   fields[2] = getEngineNames();
   if (description != null && description.length() > 0)
      fields[3] = descriptionFormat.format(new Object[]{description});
   else fields[3] = "";
   hitFormat.format(fields, out, null);
}

public String toHTML() {
   StringBuffer html = new StringBuffer();
   pasteHTML(html);
   return html.toString();
}

public static Hit merge(Hit h1, Hit h2) {

   String desc = h1.description;
   if (HTMLTools.isEmpty(desc)) desc = h2.description;
   Hit newHit = new Hit(null, h1.getURL(), h1.getTitle(), desc);
   newHit.rank = (h1.rank + h2.rank)/2;
   newHit.engines.addAll(h1.engines);
   Iterator it = h2.engines.iterator();
   while(it.hasNext()) {
      EngineAdaptor e = (EngineAdaptor) it.next();
      if (!newHit.engines.contains(e)) {
         newHit.engines.add(e);
      }
   }
   return newHit;
}

   public boolean equals(Object o) {
   if (this == o) return true;
   if (! (o instanceof Hit)) return false;
   Hit h = (Hit) o;
   return getURL().equalsIgnoreCase(h.getURL());
}
```

```
  public int compareTo(Object o) {
     if (this == o) return 0;
     Hit h = (Hit) o;
     return getTitle().compareToIgnoreCase(h.getTitle());
   }
   public int getRank() { return rank; }
   public void setRank(int newRank) { rank = newRank; }
   public int getDescriptionWeight() { return descriptionWeight; }
   public void setDescriptionWeight(int newWeight) { descriptionWeight = newWeight;
}
   public static void mergeHits(ArrayList toList, ArrayList fromList) {
     boolean toMap[] = new boolean[toList.size()];
     for(int i=0,n=fromList.size(); i<n; i++) {

        Hit fromHit = (Hit) fromList.get(i);
        int index = toList.indexOf(fromHit);
        if (index != -1 && index < toMap.length && !toMap[index]) {
          Hit oldHit = (Hit)toList.get(index);
          Hit newHit = merge(oldHit, fromHit);
          toList.set(index, newHit);
          toMap[index] = true;
        } else {
          toList.add(fromHit);
        }

     }
   }

   public static Comparator getRankComparator() {
     return new RankComparator();
   }

   static class RankComparator implements Comparator {
     public boolean equals(Object obj) {
        return RankComparator.this == obj;
     }

     public int compare(Object o1, Object o2) {
        int diff = ((Hit) o1).getRank() - ((Hit) o2).getRank();
        if (diff != 0) return diff;
        else return ((Hit)o1).compareTo(o2);
     }

   }
}
```

7) HTMLTools.java

```
package kwokchan.isearch;

import gnu.regexp.RE;
import gnu.regexp.REMatch;
import gnu.regexp.UncheckedRE;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.URLConnection;
import java.util.HashMap;

public final class HTMLTools {

   public static boolean isEmpty(String s) {
      return (s == null) || s.length() == 0;
   }

   public static String nvl(String s) {
      return (s == null) ? "" : s;
   }

   public static int indexOfIgnoreCase(String content, String re, int position) {

      if (re.length() == 0) return -1;
      if (position < 0) position = 0;
      char firstUpper = Character.toUpperCase(re.charAt(0));
      char firstLower = Character.toLowerCase(firstUpper);
      int pos = position;

      while (pos < content.length()) {
         int lowerPos = content.indexOf(firstLower, pos);
         int upperPos = content.indexOf(firstUpper, pos);
         int newPos = Math.min((lowerPos >= 0) ? lowerPos : upperPos, (upperPos >= 0)
? upperPos : lowerPos);
         if (newPos == -1) return -1;
         pos = newPos;
         if (content.regionMatches(true, pos, re, 0, re.length())) return pos;
         pos++;
      }
      return -1;
   }
   static RE commentRE = new UncheckedRE("<!--.*?-->",
RE.REG_DOT_NEWLINE);
```

```
  public static String removeComments(String html) {
    if (isEmpty(html)) return "";
    try {

      int open, close, offset = 0;
      if ((open = html.indexOf("<!--")) == -1) return html;
      StringBuffer b = new StringBuffer(html.length());
      int htmlLength = html.length();
      while(open >= 0) {
        b.append(html.substring(offset, open));
        close = html.indexOf("-->", open+4);
        if (close == -1) {
          offset = htmlLength;
          break;
        }
        offset = close+3;
        open = html.indexOf("<!--", offset);
      }
      if (offset < htmlLength)
        b.append(html.substring(offset));
      return b.toString();

    } catch (Exception ex) {
      ex.printStackTrace();
      return html;
    }
  }

  static RE tagRE = new UncheckedRE("<[^>]+>", RE.REG_DOT_NEWLINE);

    public static String removeTags(String html) {
    if (isEmpty(html)) return "";
    try {
      String newText = tagRE.substituteAll(html, "");
      return newText;

    } catch (Exception ex) {
      ex.printStackTrace();
      return html;
    }
  }


  static RE scriptRE = new UncheckedRE("<script.*?/script>",
RE.REG_DOT_NEWLINE);
```

```
  public static String removeScripts(String html) {
    if (isEmpty(html)) return "";
    try {
      String newText = scriptRE.substituteAll(html, "");
      return newText;

    } catch (Exception ex) {
      ex.printStackTrace();
      return html;
    }
  }

  static RE bodyRE = new UncheckedRE("<body[^>]*>(.*)</body>",
RE.REG_ICASE|RE.REG_DOT_NEWLINE);

  public static String getBody(String html) {
    if (isEmpty(html)) return "";
    try {

      int bodyOpen = html.indexOf("<body");
      if (bodyOpen == -1) bodyOpen = html.indexOf("<BODY");
      if (bodyOpen >= 0)  {
        int bodyClose = html.indexOf("</body>", bodyOpen);
        if (bodyClose == -1) bodyClose = html.indexOf("</BODY>", bodyOpen);
        if (bodyClose > bodyOpen) {
          return html.substring(bodyOpen, bodyClose);
        }
      }


      REMatch match = bodyRE.getMatch(html);
      if (match == null) return html;
      String body = match.toString(1);
      return body;

    } catch (Exception ex) {
      ex.printStackTrace();
      return html;
    }


  }
```

```
    public static String readContent(URLConnection http, long deadline) throws
IOException {

      ByteArrayOutputStream bs = new ByteArrayOutputStream();
      InputStream is = http.getInputStream();
      byte[] buffer = new byte[1024];
      int count;
      while((count = is.read(buffer)) > 0) {
        bs.write(buffer, 0, count);
        if (deadline > 0 && System.currentTimeMillis() >= deadline) break;
      }
      String content = bs.toString();
      return content;
    }

    public static int atoi(String s) {
      try {
        return Integer.parseInt(s);
      } catch (Exception ex) {
        return 0;
      }
    }


    private static RE unicodeRE = new UncheckedRE("&#(\\d+);");
    private static RE specialCharRE = new UncheckedRE("&([a-zA-Z]+);");
    private static HashMap specialCharMap = new HashMap();

    private static String latin =
      "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ_$";
    private static String spaces =
      " \t\n\r\u00ff";
    private static final String fourDots = "....";
    private static final String fourQuestions = "????";
    private static final String qdot = "?.";

    public static boolean isForeign(String html) {

      if (html.indexOf(fourDots) != -1) return true;
      if (html.indexOf(fourQuestions) != -1) return true;

      int offset = 0;
      REMatch match;
      StringBuffer s = new StringBuffer(html);
```

```
    while( (match=unicodeRE.getMatch(s, offset)) != null) {
      int code = atoi(match.toString(1));
      if (code > 255) {
        return true;
      }
      char c = (char) code;
      s.replace(match.getStartIndex(), match.getEndIndex(), String.valueOf(c));
      offset = match.getStartIndex() + 1;
    }

    offset = 0;
    while( (match=specialCharRE.getMatch(s, offset)) != null) {
      String seq = match.toString(1);
      String c = (String) specialCharMap.get(seq.toLowerCase());
      if (c == null) c = "a";
      s.replace(match.getStartIndex(), match.getEndIndex(), c);
      offset = match.getStartIndex() + 1;
    }

    int foreignCount = 0;
    int latinCount   = 0;
    int punctCount   = 0;
    int totalCount   = 0;

    for(int i=0,n=s.length(); i<n; i++) {
      char c = s.charAt(i);
      if (spaces.indexOf(c) >= 0) continue;
      else if (((int)c) > 128) foreignCount++;
      else if (latin.indexOf(c) >= 0) latinCount++;
      else if (qdot.indexOf(c) >= 0) punctCount++;
      totalCount++;
    }

    if (foreignCount >= 5 && foreignCount > latinCount &&
      foreignCount > (totalCount/2))
      return true;

    if (punctCount == totalCount)
      return true;
    if (totalCount >= 12 && punctCount > (totalCount / 4 * 3))
      return true;

    return false;
  }
```

```
   private static final RE untitledRE = new
UncheckedRE("\\W*untitled(\\s+document)?\\W*", RE.REG_ICASE);
   private static final RE noTitleRE = new UncheckedRE("\\W*no\\s*title\\W*",
RE.REG_ICASE);
   private static final RE cachedRE = new UncheckedRE("\\W*cached\\W*",
RE.REG_ICASE);

   private static final String badFirstChar = "01234567890?.";

   public static boolean isGarbage(String html) {

      if (isEmpty(html)) return true;
      if (html.indexOf(fourDots) != -1) return true;
      if (html.indexOf(fourQuestions) != -1) return true;
      char c = html.charAt(0);
      if (badFirstChar.indexOf(c) != -1) return true;

      if (untitledRE.isMatch(html)) return true;
      if (noTitleRE.isMatch(html)) return true;
      if (cachedRE.isMatch(html)) return true;

      int totalCount = 0, punctCount = 0;
      for(int i=0,n=html.length(); i<n; i++) {
         c = html.charAt(i);
         if (spaces.indexOf(c) != -1) continue;
         totalCount++;
         if (qdot.indexOf(c) != -1) punctCount++;
      }
      if (punctCount >= totalCount-2)
         return true;


      return false;
   }


   static {
      specialCharMap.put("amp", "&");
      specialCharMap.put("quot", "\"");
      specialCharMap.put("lt", "<");
      specialCharMap.put("gt", ">");
   }

}
```

8) MutableBoolean.java

```
package kwokchan.isearch;

public final class MutableBoolean {

 boolean value;
 public MutableBoolean() {}
 public MutableBoolean(boolean b) { value = b;}
 public boolean getValue() { return value;}
 public void setValue(boolean b) { this.value = b; }

}
```

9) Searcher.java

```
package kwokchan.isearch;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Iterator;

public class Searcher implements Runnable {
   public static final int SORT_BY_RELEVANCE = 0;
   public static final int SORT_ALPHABETICALLY = 1;
   private MutableBoolean interuptor = new MutableBoolean();
   private boolean completed;
   private EngineSet engines;
   private EngineRequest request;
   private int sortOrder;
   private StringBuffer stats;
   private ArrayList results;

   public Searcher(EngineSet engines, EngineRequest request, int sortOrder) {
      this.engines = engines;
      this.request = request;
      this.sortOrder = sortOrder;
      this.stats = new StringBuffer();
      this.results = null;
   }
   public ArrayList getResults() {
      return results;
   }
```

```
  public boolean isCompleted() { return completed; }
  public boolean isInterrupted() { return interuptor.getValue(); }
  public void interrupt() {
    interuptor.setValue(true);
  }
  public String getStats() {
    return stats.toString();
  }
  public void run() {
    long t1 = System.currentTimeMillis();
    completed = false;
    interuptor.setValue(false);
    ArrayList results = new ArrayList();
    this.results = results;
    stats.setLength(0);
    results.clear();
    final EngineAdaptor[] engines = this.engines.getAllEngines();
    int n = engines.length;
    final EngineThread[] workers = new EngineThread[n];
    final Thread[] threads = new Thread[n];
    final Object synchronizer = new Object();

    for(int i=0; i<n; i++) {
      EngineThread worker = new EngineThread(engines[i], request, interuptor,
synchronizer);
      workers[i] = worker;
      threads[i] = new Thread(worker);
    }
    for(int i=0; i<n; i++) {
      threads[i].start();
    }

    while(true) {
      try {
        synchronized(synchronizer) {
          synchronizer.wait(1000);
        }
        int leftToGo = 0;
        for(int i=0; i<n; i++) {
          if (!workers[i].isReady()) leftToGo++;
        }
        if (leftToGo == 0) break;
        } catch (Throwable ex) {
      }
    }
```

```
    for(int i=0; i<n; i++) {
      stats.append(workers[i].getStats());
      try {
        Hit.mergeHits(results, workers[i].getResults());
      } catch (Throwable ex) {
        ex.printStackTrace();
      }
    }
    int foreignRemoved = 0;
    if (request.isFlagSet(EngineRequest.FOREIGN_FILTER)) {
      Iterator it = results.iterator();
      while(it.hasNext()) {
        Hit hit = (Hit) it.next();
        if (HTMLTools.isForeign(hit.getTitle())) {
          it.remove();
          foreignRemoved++;
        }
      }
    }

    int garbageRemoved = 0;
    if (request.isFlagSet(EngineRequest.GARBAGE_FILTER)) {
      Iterator it = results.iterator();
      while(it.hasNext()) {
        Hit hit = (Hit) it.next();
        if (HTMLTools.isGarbage(hit.getTitle())) {
          it.remove();
          garbageRemoved++;
        }
      }
    }

    if (sortOrder == SORT_ALPHABETICALLY) {
      Collections.sort(results);
    } else if (sortOrder == SORT_BY_RELEVANCE) {
      Collections.sort(results, Hit.getRankComparator());
    }
    if (foreignRemoved > 0) {
      stats.append("<B>"+foreignRemoved+"</B> result(s) were removed as
foreign<BR>\n");
    }
    if (garbageRemoved > 0) {
      stats.append("<B>"+garbageRemoved+"</B> result(s) were removed as
\"garbage\"<BR>\n");
    }
```

```
      if (results.size() > 0) {
         long totalTime = System.currentTimeMillis() - t1;
         stats.append("Total Results: <B>"+results.size()+"</B> ("+totalTime+"
ms)<BR>\n");
      }
      completed = true;

   }
}
```

10) Config.java

```
package kwokchan.isearch.gui;
import java.awt.Rectangle;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.net.URLDecoder;
import java.net.URLEncoder;
import java.util.Enumeration;
import java.util.Hashtable;
import java.util.StringTokenizer;

public final class Config extends Hashtable {

   static Config instance = new Config();
   static File file = new File(System.getProperty("user.home"), ".isearch");
   public static String DEFAULT_ENGINES = "google,hotbot,";
   private Config() {
   }
   public static boolean isWindows() {
      String os = System.getProperty("os.name");
      return (os != null) && os.startsWith("Win");
   }

   public static boolean isDefaultEngine(String ID) {
      return DEFAULT_ENGINES.indexOf(ID+",") >= 0;
   }

   public static void setString(String name, String value) {
      instance.put(name, value);
   }
```

```
public static void setInt(String name, int value) {
    instance.put(name, Integer.toString(value));
  }

  public static void setBoolean(String name, boolean value) {
    instance.put(name, value ? "1" : "0");
  }

  public static void setGeometry(String name, Rectangle value) {
    instance.put(name,
      String.valueOf(value.width) + "/" +
      String.valueOf(value.height) + "/" +
      String.valueOf(value.x) + "/" +
      String.valueOf(value.y)
    );
  }

  public static boolean getBoolean(String name, boolean defaultValue) {
    String value = (String) instance.get(name);
    if (isEmpty(value)) return defaultValue;
    return atoi(value) != 0;
  }

  public static String getString(String name, String defaultValue) {
    String value = (String) instance.get(name);
    if (isEmpty(value)) value = defaultValue;
    return value;
  }
  public static Rectangle getGeometry(String name, Rectangle defaultValue) {
    Rectangle value;
    try {
      String s = (String) instance.get(name);
      if (s == null) return defaultValue;
      StringTokenizer t = new StringTokenizer(s, "/");
      if (t.countTokens() != 4) throw new Exception("invalid geometry: "+s);
      int width  = Integer.parseInt(t.nextToken());
      int height = Integer.parseInt(t.nextToken());
      int x      = Integer.parseInt(t.nextToken());
      int y      = Integer.parseInt(t.nextToken());
      value = new Rectangle(x, y, width, height);
    } catch (Exception ex) {
      value = defaultValue;
    }
    return value;
  }
```

```
public static int getInt(String name, int defaultValue) {
   String value = (String) instance.get(name);
   if (isEmpty(value)) return defaultValue;
   return atoi(value);
}

public static boolean isEmpty(String s) {
   return (s == null || s.length() == 0);
}

public static int atoi(String s) {
   try {
      return Integer.parseInt(s);
   } catch (Exception ex) {
      return 0;
   }
}

public static void load() {
  try {
     if (!file.canRead()) return;
     instance.clear();
     FileInputStream is = new FileInputStream(file);
     ByteArrayOutputStream ba = new ByteArrayOutputStream();
     byte[] buffer = new byte[256]; int count;
     while((count = is.read(buffer)) > 0) {
        ba.write(buffer, 0, count);
     }
     is.close();
     StringTokenizer lines = new StringTokenizer(ba.toString(), "\n");
     while(lines.hasMoreTokens()) {
        String line = lines.nextToken();
        int eq = line.indexOf('=');
        if (eq <= 0) continue;
        String key   = URLDecoder.decode( line.substring(0, eq).trim() );
        String value = URLDecoder.decode( line.substring(eq+1).trim() );
        instance.put(key, value);
     }

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

```
public static void save() {

    try {
       FileWriter fw = new FileWriter(file);
       Enumeration keys = instance.keys();
       while(keys.hasMoreElements()) {
          String key   = (String) keys.nextElement();
          String value = getString(key, null);
          if (value == null) continue;
          fw.write(URLEncoder.encode(key));
          fw.write("=");
          fw.write(URLEncoder.encode(value));
          fw.write("\n");
       }
       fw.close();

    } catch (Exception ex) {
       ex.printStackTrace();
    }
  }

}
```

11) ConfigWindow.java

```
package kwokchan.isearch.gui;
import java.awt.*;
import java.awt.event.ActionEvent;
import javax.swing.*;
import kwokchan.isearch.EngineAdaptor;
import kwokchan.isearch.EngineRegistry;

public class ConfigWindow extends JDialog implements ISearchConstants {

   private static Object[] resultCounts = { "10", "20", "40", "100", "200" };
   private static ComboBoxModel resultCountModel = new
DefaultComboBoxModel(resultCounts);

   private static Object[] timeoutValues = { "0", "5", "10", "20", "60" };
   private static ComboBoxModel timeoutModel = new
DefaultComboBoxModel(timeoutValues);

   EngineBox engineBoxes[];
   private boolean approved;
```

```
  JPanel mainPanel = new JPanel();
  GridBagLayout gridBagLayout2 = new GridBagLayout();
  GridBagLayout gridBagLayout3 = new GridBagLayout();
  JButton okButton = new JButton();
  JButton cancelButton = new JButton();
  JPanel buttonPanel = new JPanel();
  JPanel contentPanel = new JPanel();
  JLabel resultCountCaption = new JLabel();
  FlowLayout flowLayout1 = new FlowLayout();
  JComboBox resultCountField = new JComboBox();
  JLabel enginesCaption = new JLabel();
  GridBagLayout gridBagLayout1 = new GridBagLayout();
  JScrollPane enginesScroller = new JScrollPane();
  JPanel enginesPanel = new JPanel();
  GridBagLayout gridBagLayout4 = new GridBagLayout();
  JPanel timeoutPanel = new JPanel();
  GridBagLayout gridBagLayout5 = new GridBagLayout();
  JLabel timeoutCaption = new JLabel();
  JComboBox timeoutField = new JComboBox();
  JLabel secondsCaption = new JLabel();
  JPanel browserPanel = new JPanel();
  GridBagLayout gridBagLayout6 = new GridBagLayout();
  JRadioButton defaultBrowserButton = new JRadioButton();
  ButtonGroup browserTypeButtonGroup = new ButtonGroup();
  JRadioButton customBrowserButton = new JRadioButton();
  JTextField commandLineField = new JTextField();
  JLabel commandLineComment = new JLabel();
  JCheckBox foreignFilterButton = new JCheckBox();
  JCheckBox garbageFilterButton = new JCheckBox();

  public ConfigWindow(Frame frame, String title) {
    super(frame, title, true);
    try {
      jbInit();
      setupEngines();
      int count = Config.getInt(MATCH_COUNT, MATCH_COUNT_DEFAULT);
      resultCountField.setSelectedItem(String.valueOf(count));
      foreignFilterButton.setSelected(Config.getBoolean(FOREIGN_FILTER,
FOREIGN_FILTER_DEFAULT));
      garbageFilterButton.setSelected(Config.getBoolean(GARBAGE_FILTER,
GARBAGE_FILTER_DEFAULT));
      int timeout = Config.getInt(SEARCH_TIMEOUT,
SEARCH_TIMEOUT_DEFAULT);
      timeoutField.setSelectedItem(String.valueOf(timeout));
```

```
        boolean customBrowser = (!Config.isWindows()) ||
          Config.getBoolean(USE_CUSTOM_BROWSER,
USE_CUSTOM_BROWSER_DEFAULT);
        String browseCommand = Config.getString(BROWSE_COMMAND, "");

        defaultBrowserButton.setSelected(!customBrowser );
        customBrowserButton.setSelected(customBrowser);
        defaultBrowserButton.setEnabled(Config.isWindows());
        commandLineField.setText(browseCommand);
        commandLineField.setEnabled(customBrowserButton.isSelected());
        pack();
        setLocationRelativeTo(frame);
      }
      catch(Exception ex) {
        ex.printStackTrace();
      }
    }
    public boolean isApproved() {
      return approved;
    }

    private void setupEngines() {
      EngineAdaptor[] engines = EngineRegistry.getAllAdaptors();

      int n = engines.length;
      engineBoxes = new EngineBox[n];
      Insets zeroInsets = new Insets(0, 0, 0, 0);
      enginesPanel.removeAll();
      for(int i=0; i<n; i++) {
        EngineBox box = new EngineBox(engines[i]);
        String id = engines[i].getID();
        engineBoxes[i] = box;
        Object constraints = new GridBagConstraints(0, i, 1, 1, 1.0, 0.0,
          GridBagConstraints.NORTHWEST, GridBagConstraints.HORIZONTAL,
zeroInsets, 0, 0);
        enginesPanel.add(box, constraints);
        box.setSelected(
          Config.getBoolean("include."+id, Config.isDefaultEngine(id))
        );
      }
      enginesPanel.add(new JLabel(" "),  new GridBagConstraints(0, n, 1, 1, 1.0, 1.0
        ,GridBagConstraints.CENTER, GridBagConstraints.BOTH, zeroInsets, 0, 0));
      enginesPanel.doLayout();
    }
```

```
  private boolean validate(boolean silent) {

    int engineCount = 0;
    for(int i=0,n=engineBoxes.length; i<n; i++) {
      if (engineBoxes[i].isSelected()) engineCount++;
    }
    if (engineCount == 0) {
      if (!silent) JOptionPane.showMessageDialog(this, "Please select at least one
engine");
      return false;
    }

    int resultCount = Config.atoi((String) resultCountField.getSelectedItem());
    if (resultCount <= 0) {
      if (!silent) JOptionPane.showMessageDialog(this, "Please enter valid result
count");
      return false;
    }

    int timeout = Config.atoi((String) timeoutField.getSelectedItem());
    if (timeout < 0) {
      if (!silent) JOptionPane.showMessageDialog(this, "Please enter valid timeout
value");
      return false;
    }

    return true;
  }

  private void applyChanges() {
    for(int i=0,n=engineBoxes.length; i<n; i++) {
      engineBoxes[i].apply();
    }
    Config.setInt(MATCH_COUNT, Config.atoi((String)
resultCountField.getSelectedItem()));
    Config.setInt(SEARCH_TIMEOUT, Config.atoi((String)
timeoutField.getSelectedItem()));
    Config.setBoolean(FOREIGN_FILTER, foreignFilterButton.isSelected());
    Config.setBoolean(GARBAGE_FILTER, garbageFilterButton.isSelected());
    Config.setBoolean(USE_CUSTOM_BROWSER,
customBrowserButton.isSelected());
    Config.setString(BROWSE_COMMAND, commandLineField.getText());

    Config.save();
  }
```

```
void jbInit() throws Exception {
  mainPanel.setLayout(gridBagLayout3);
  this.getContentPane().setLayout(gridBagLayout2);
  okButton.setToolTipText("");
  okButton.setText("OK");
  okButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
      okButton_actionPerformed(e);
    }
  });
  cancelButton.setText("Cancel");
  cancelButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
      cancelButton_actionPerformed(e);
    }
  });
  buttonPanel.setLayout(flowLayout1);
  resultCountCaption.setText("Number of results per engine:");
  resultCountField.setMinimumSize(new Dimension(60, 20));
  resultCountField.setPreferredSize(new Dimension(60, 20));
  resultCountField.setEditable(true);
  resultCountField.setModel(resultCountModel);
  enginesCaption.setText("Active Engines:");
  contentPanel.setLayout(gridBagLayout1);
  enginesPanel.setLayout(gridBagLayout4);
  enginesScroller.setMinimumSize(new Dimension(300, 100));
  enginesScroller.setPreferredSize(new Dimension(300, 100));
  timeoutPanel.setLayout(gridBagLayout5);
  timeoutCaption.setText("Engine Timeout (0=wait forever):");
  secondsCaption.setText("seconds");
  timeoutField.setMinimumSize(new Dimension(60, 20));
  timeoutField.setPreferredSize(new Dimension(60, 20));
  timeoutField.setEditable(true);
  timeoutField.setModel(timeoutModel);
  browserPanel.setLayout(gridBagLayout6);
  defaultBrowserButton.setSelected(true);
  defaultBrowserButton.setText("Use Default Browser (Windows only)");
  defaultBrowserButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
      defaultBrowserButton_actionPerformed(e);
    }
  });
  customBrowserButton.setText("Specify command line:");
  customBrowserButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
```

```
        customBrowserButton_actionPerformed(e);
      }
    });
    commandLineComment.setFont(new java.awt.Font("Dialog", 0, 10));
    commandLineComment.setText("%1 will be replaced by URL");
    foreignFilterButton.setToolTipText("");
    foreignFilterButton.setText("Remove foreign (non-English) titles");
    garbageFilterButton.setText("Aggressively remove \"garbage\" titles");
    getContentPane().add(mainPanel,  new GridBagConstraints(0, 0, 1, 1, 1.0, 1.0
,GridBagConstraints.CENTER, GridBagConstraints.BOTH, new Insets(4, 4, 4, 4), 0, 0));
    mainPanel.add(buttonPanel,      new GridBagConstraints(0, 1, 1, 1, 0.0, 0.0
  ,GridBagConstraints.EAST, GridBagConstraints.NONE, new Insets(0, 0, 0, 0), 0, 0));
    buttonPanel.add(okButton, null);
    buttonPanel.add(cancelButton, null);
    mainPanel.add(contentPanel,      new GridBagConstraints(0, 0, 1, 1, 1.0, 1.0
,GridBagConstraints.CENTER, GridBagConstraints.BOTH, new Insets(0, 0, 0, 0), 2, 2));
    contentPanel.add(resultCountCaption,   new GridBagConstraints(0, 0, 1, 1, 0.0, 0.0
  ,GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(0, 0, 0, 0), 0, 0));
    contentPanel.add(resultCountField,        new GridBagConstraints(1, 0, 1, 1, 0.0, 0.0
  ,GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(0, 4, 0, 4), 0, 0));
    contentPanel.add(enginesCaption,      new GridBagConstraints(0, 1, 1, 1, 0.0, 0.0
 ,GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(0, 0, 0, 12), 0, 0));
    contentPanel.add(enginesScroller,       new GridBagConstraints(0, 2, 2, 1, 1.0, 1.0
,GridBagConstraints.CENTER, GridBagConstraints.BOTH, new Insets(4, 0, 0, 0), 0, 0));
    contentPanel.add(timeoutPanel,          new GridBagConstraints(0, 3, 2, 1, 1.0, 0.0
  ,GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(4, 0, 4, 0), 0, 0));
    enginesScroller.getViewport().add(enginesPanel, null);
    timeoutPanel.add(timeoutCaption,    new GridBagConstraints(0, 2, 1, 1, 0.0, 0.0
,GridBagConstraints.CENTER, GridBagConstraints.NONE, new Insets(0, 0, 0, 0), 0, 0));
    timeoutPanel.add(timeoutField,       new GridBagConstraints(1, 2, 1, 1, 0.0, 0.0
,GridBagConstraints.CENTER, GridBagConstraints.NONE, new Insets(0, 4, 0, 4), 0, 0));
    timeoutPanel.add(secondsCaption,     new GridBagConstraints(2, 2, 1, 1, 0.0, 0.0
,GridBagConstraints.CENTER, GridBagConstraints.NONE, new Insets(0, 0, 0, 0), 0, 0));
    timeoutPanel.add(foreignFilterButton,    new GridBagConstraints(0, 0, 3, 1, 0.0, 0.0
  ,GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(0, 0, 0, 0), 0, 0));
    timeoutPanel.add(garbageFilterButton,  new GridBagConstraints(0, 1, 3, 1, 0.0, 0.0
  ,GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(0, 0, 0, 0), 0, 0));
    contentPanel.add(browserPanel,   new GridBagConstraints(0, 4, 2, 1, 1.0, 0.0
,GridBagConstraints.CENTER, GridBagConstraints.HORIZONTAL, new Insets(0, 0, 8,
0), 0, 0));
 browserPanel.add(defaultBrowserButton,      new GridBagConstraints(0, 0, 2, 1, 0.0, 0.0
  ,GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(0, 0, 0, 0), 0, 0));
 browserPanel.add(customBrowserButton,    new GridBagConstraints(0, 1, 1, 1, 0.0, 0.0
,GridBagConstraints.CENTER, GridBagConstraints.NONE, new Insets(0, 0, 0, 0), 0, 0));
  browserPanel.add(commandLineField,     new GridBagConstraints(1, 1, 1, 1, 1.0, 0.0
,GridBagConstraints.CENTER, GridBagConstraints.HORIZONTAL, new Insets(0, 0, 0,
0), 0, 0));
```

```
browserPanel.add(commandLineComment, new GridBagConstraints(1, 2, 1, 1, 0.0, 0.0
    ,GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(0, 0, 0, 0), 0, 0));
      browserTypeButtonGroup.add(defaultBrowserButton);
      browserTypeButtonGroup.add(customBrowserButton);
    }
    void okButton_actionPerformed(ActionEvent e) {
      if (!validate(false)) return;
      applyChanges();
      setVisible(false);
    }
    void cancelButton_actionPerformed(ActionEvent e) {
      setVisible(false);
    }
    void defaultBrowserButton_actionPerformed(ActionEvent e) {
      commandLineField.setEnabled(customBrowserButton.isSelected());
    }
    void customBrowserButton_actionPerformed(ActionEvent e) {
      commandLineField.setEnabled(customBrowserButton.isSelected());
    }


}
```

12) EngineBox.java

```
package kwokchan.isearch.gui;
import java.awt.Dimension;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import javax.swing.Icon;
import javax.swing.JCheckBox;
import javax.swing.JLabel;
import javax.swing.JPanel;
import kwokchan.isearch.EngineAdaptor;

public class EngineBox extends JPanel {

    private EngineAdaptor engine;
    private String configName;

    JCheckBox enabledButton = new JCheckBox();
    GridBagLayout gridBagLayout1 = new GridBagLayout();
    JLabel logoLabel = new JLabel();
```

```
    public EngineBox(EngineAdaptor engine) {
        try {
            jbInit();
            this.engine = engine;
            configName = "include."+engine.getID();
            Icon icon = engine.getIcon(0);
            if (icon != null) logoLabel.setIcon(icon);
            logoLabel.setText(engine.getEngineURL());
            enabledButton.setSelected(Config.getBoolean(configName, true));
        }
        catch(Exception ex) {
            ex.printStackTrace();
        }
    }

    public boolean isSelected() {
        return enabledButton.isSelected();
    }

    public void setSelected(boolean b) {
        enabledButton.setSelected(b);
    }

    public void apply() {
        Config.setBoolean(configName, isSelected());
    }


    void jbInit() throws Exception {
        this.setLayout(gridBagLayout1);
        logoLabel.setMaximumSize(new Dimension(1, 32));
        logoLabel.setMinimumSize(new Dimension(75, 32));
        logoLabel.setPreferredSize(new Dimension(75, 32));
        logoLabel.setIconTextGap(8);
        logoLabel.setText(" ");
        this.add(enabledButton,     new GridBagConstraints(0, 0, 1, 1, 0.0, 0.0
            ,GridBagCraints.WEST, GridBagConstraints.NONE, new Insets(2, 4, 2, 4), 0, 0));
        this.add(logoLabel,  new GridBagConstraints(1, 0, 1, 1, 1.0, 0.0
       ,GridBagConstraints.CENTER, GridBagConstraints.HORIZONTAL, new Insets(0,
0, 0, 4), 0, 0));
    }
}
```

13) HistoryComboBox.java

```
package kwokchan.isearch.gui;

import javax.swing.ComboBoxEditor;
import javax.swing.DefaultComboBoxModel;
import javax.swing.JComboBox;
public class HistoryComboBox extends JComboBox {

   DefaultComboBoxModel model;
    public HistoryComboBox() {
      model = new DefaultComboBoxModel();
      setModel(model);
   }

   public synchronized void commit() {
      String item = (String) getSelectedItem();
      if (item != null && (item=item.trim()).length() > 0) {
         String selection = (String) model.getSelectedItem();
         int index = model.getIndexOf(item);
         if (index != -1) {
            model.removeElementAt(index);
         }
         model.insertElementAt(item, 0);
         if (model.getSize() > 10) {
            for(int i=10,n=model.getSize(); i<n; i++)
               model.removeElementAt(i);
         }
         model.setSelectedItem(selection);
      }
   }
   public String getText() {
      ComboBoxEditor editor = getEditor();
      String s;
      if (editor != null) s = (String) editor.getItem();
      else s = (String) getSelectedItem();
      if (s == null) s = "";
      return s;
   }
   public void setText(String s) {
      ComboBoxEditor editor = getEditor();
      if (s == null) s = "";
      if (editor != null) editor.setItem(s);
      else setSelectedItem(s);
   }
}
```

14) Isearch.java

```
package kwokchan.isearch.gui;
import java.awt.Dimension;
import java.awt.Rectangle;
import java.awt.Toolkit;
import javax.swing.UIManager;
public class Isearch {
   public static void main(String[] args) {
      if (Config.isWindows()) {
         try {

UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel"
);
         } catch (Throwable ex) {
         }
      }

      Config.load();
      MainFrame frame = new MainFrame();
      frame.pack();
      Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
      if (Config.isWindows()) screenSize.height -= 24;
      Rectangle bounds = Config.getGeometry(ISearchConstants.FRAME_GEOMETRY,
null);
      if (bounds == null || bounds.isEmpty()) {
         Dimension frameSize = new Dimension(screenSize);
         frameSize.width = frameSize.width * 2 / 3;
         frameSize.height = frameSize.height * 2 / 3;
         if (frameSize.height > screenSize.height) frameSize.height = screenSize.height;
         if (frameSize.width > screenSize.width) frameSize.width = screenSize.width;
         bounds = new Rectangle(
            (screenSize.width - frameSize.width) / 2,
            (screenSize.height - frameSize.height) / 2,
            frameSize.width,
            frameSize.height
         );
      }
      frame.setBounds(bounds);
      frame.setVisible(true);
      }
 }
```

15) IsearchConstants.java

```
package kwokchan.isearch.gui;
public interface ISearchConstants {
   public static final String MATCH_COUNT = "perEngineResults";
   public static final int MATCH_COUNT_DEFAULT = 40;
   public static final String SEARCH_TIMEOUT = "searchTimeout";
   public static final int SEARCH_TIMEOUT_DEFAULT = 0;
   public static final String FRAME_GEOMETRY = "frameSize";
   public static final String FOREIGN_FILTER = "foreignFilter";
   public static final boolean FOREIGN_FILTER_DEFAULT = false;
   public static final String GARBAGE_FILTER = "garbageFilter";
   public static final boolean GARBAGE_FILTER_DEFAULT = false;
   public static final String USE_CUSTOM_BROWSER = "useCustomBrowser";
   public static final boolean USE_CUSTOM_BROWSER_DEFAULT = false;
   public static final String BROWSE_COMMAND = "browseCommand";
   public static final String ALPHABETICAL_SORT = "alphaSort";
   public static final boolean ALPHABETICAL_SORT_DEFAULT = false;
}
```

16) MainFrame.java

```
package kwokchan.isearch.gui;
import gnu.regexp.RE;
import gnu.regexp.UncheckedRE;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ComponentEvent;
import java.awt.event.KeyEvent;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;
import java.io.File;
import java.io.FileWriter;
import java.net.URL;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Vector;
import javax.swing.*;
import javax.swing.event.HyperlinkEvent;
import kwokchan.isearch.*;

public class MainFrame extends JFrame implements ISearchConstants {
   static boolean debug;
   private static java.io.PrintStream Debug = System.out;
   private static final String TITLE = "Kwok Chan Search Engine";
```

```
  private File previousFile;
  private static final String BR = "<BR>\n";

  JLabel statusBar = new JLabel();
  JPanel mainPanel = new JPanel();
  GridBagLayout gridBagLayout1 = new GridBagLayout();
  HistoryComboBox patternField = new HistoryComboBox();
  JLabel inputFieldCaption = new JLabel();
  JScrollPane resultPaneScroll = new JScrollPane();
  JTextPane resultPane = new JTextPane();
  GridBagLayout gridBagLayout2 = new GridBagLayout();
  JButton searchButton = new JButton();
  JMenuBar mainMenu = new JMenuBar();
  JMenu menuFile = new JMenu();
  JMenu menuTools = new JMenu();
  JMenuItem menuFileExit = new JMenuItem();
  JMenuItem menuToolsConfiguration = new JMenuItem();
  JPanel optionsPanel = new JPanel();
  GridBagLayout gridBagLayout3 = new GridBagLayout();
  JRadioButton relevanceButton = new JRadioButton();
  JRadioButton alphabeticalButton = new JRadioButton();
  ButtonGroup sortButtonGroup = new ButtonGroup();
  JLabel sortCaption = new JLabel();
  JLabel actionLabel = new JLabel();

  public MainFrame() {
    try {
      jbInit();
      mainPanel.setPreferredSize(new Dimension(600, 300));
      boolean alphaSort = Config.getBoolean(ALPHABETICAL_SORT,
ALPHABETICAL_SORT_DEFAULT);
      if (alphaSort) alphabeticalButton.setSelected(true);
      else relevanceButton.setSelected(true);
      URL iconURL = getClass().getClassLoader().getResource("isearch.gif");
      setIconImage(Toolkit.getDefaultToolkit().getImage(iconURL));
      actionLabel.setText(" ");
      resultPane.addPropertyChangeListener("page", new PropertyChangeListener() {
        public void propertyChange(PropertyChangeEvent evt) {
          pageChanged(evt);
        }
      });

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

```
private void jbInit() throws Exception {

    this.setTitle(TITLE);
    statusBar.setText(" ");
    this.getContentPane().setLayout(gridBagLayout2);
    mainPanel.setLayout(gridBagLayout1);
    inputFieldCaption.setDisplayedMnemonic('0');
    inputFieldCaption.setText("Search For:");
    resultPane.setEditable(false);
    resultPane.addHyperlinkListener(new javax.swing.event.HyperlinkListener() {
      public void hyperlinkUpdate(HyperlinkEvent e) {
        resultPane_hyperlinkUpdate(e);
      }
    });
    searchButton.setMnemonic('S');
    searchButton.setText("Search");
    searchButton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        performSearch();
      }
    });
    this.setJMenuBar(mainMenu);
    this.addComponentListener(new java.awt.event.ComponentAdapter() {
      public void componentResized(ComponentEvent e) {
        this_componentResized(e);
      }
    });
    menuFile.setMnemonic('F');
    menuFile.setText("File");
    menuTools.setMnemonic('T');
    menuTools.setText("Tools");
    menuFileExit.setText("Exit");
    menuFileExit.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        menuFileExit_actionPerformed(e);
      }
    });
    menuToolsConfiguration.setText("Configuration");
    menuToolsConfiguration.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        menuToolsConfiguration_actionPerformed(e);
      }
    });
    patternField.setEditable(true);
    optionsPanel.setLayout(gridBagLayout3);
    relevanceButton.setSelected(true);
    relevanceButton.setText("By Relevance");
```

```
    relevanceButton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        relevanceButton_actionPerformed(e);
      }
    });
    alphabeticalButton.setText("Alphabetically");
    alphabeticalButton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        alphabeticalButton_actionPerformed(e);
      }
    });
    sortCaption.setText("Sort:");
    actionLabel.setMaximumSize(new Dimension(100, 16));
    actionLabel.setMinimumSize(new Dimension(100, 16));
    actionLabel.setPreferredSize(new Dimension(100, 16));
    actionLabel.setHorizontalAlignment(SwingConstants.RIGHT);
    actionLabel.setText("Ready");
    this.getContentPane().add(statusBar,    new GridBagConstraints(0, 2, 1, 1, 1.0, 0.0
,GridBagConstraints.WEST, GridBagConstraints.HORIZONTAL, new Insets(2, 4, 2, 4),
0, 0));
    this.getContentPane().add(mainPanel,    new GridBagConstraints(0, 0, 2, 1, 1.0, 1.0
,GridBagConstraints.CENTER, GridBagConstraints.BOTH, new Insets(2, 4, 0, 4), 0, 0));
    mainPanel.add(patternField,       new GridBagConstraints(1, 0, 1, 1, 1.0, 0.0
,GridBagConstraints.WEST, GridBagConstraints.HORIZONTAL, new Insets(0, 4, 0,
4), 0, 0));
    mainPanel.add(resultPaneScroll,       new GridBagConstraints(0, 2, 3, 1, 1.0, 1.0
,GridBagConstraints.CENTER, GridBagConstraints.BOTH, new Insets(0, 0, 0, 0), 0, 0));
    mainPanel.add(searchButton,    new GridBagConstraints(2, 0, 1, 1, 0.0, 0.0
,GridBagConstraints.CENTER, GridBagConstraints.NONE, new Insets(0, 0, 0, 0), 0, 0));
    mainPanel.add(inputFieldCaption,   new GridBagConstraints(0, 0, 1, 1, 0.0, 0.0
  ,GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(0, 0, 0, 0), 0, 0));
    mainPanel.add(optionsPanel,    new GridBagConstraints(0, 1, 3, 1, 0.0, 0.0
  ,GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(0, 0, 0, 0), 0, 0));
    this.getContentPane().add(actionLabel,    new GridBagConstraints(1, 2, 1, 1, 0.0, 0.0
  ,GridBagConstraints.EAST, GridBagConstraints.NONE, new Insets(0, 2, 0, 4), 0, 0));
    resultPaneScroll.getViewport().add(resultPane, null);
    mainMenu.add(menuFile);
    mainMenu.add(menuTools);
    menuFile.add(menuFileExit);
    menuTools.add(menuToolsConfiguration);
    optionsPanel.add(relevanceButton,   new GridBagConstraints(1, 0, 1, 1, 0.0, 0.0
  ,GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(0, 0, 0, 0), 0, 0));
    optionsPanel.add(alphabeticalButton,    new GridBagConstraints(2, 0, 1, 1, 0.0, 0.0
  ,GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(0, 0, 0, 0), 0, 0));
    sortButtonGroup.add(relevanceButton);
    sortButtonGroup.add(alphabeticalButton);
```

```
    optionsPanel.add(sortCaption,    new GridBagConstraints(0, 0, 1, 1, 0.0, 0.0
  ,GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(0, 0, 0, 8), 0, 0));
  }

  private EngineSet createEngineSet() {
    Vector selectedIDs = new Vector();
    String[] allIDs = EngineRegistry.getEngineIDs();
    for(int i=0,n=allIDs.length; i<n; i++) {
      String id = allIDs[i];
      if (Config.getBoolean("include."+id, Config.isDefaultEngine(id))) {
        selectedIDs.add(id);
      }
    }
    String[] ids = (String[]) selectedIDs.toArray(new String[selectedIDs.size()]);
    return new EngineSet(ids);
  }
 long renderStartTime;

  void performSearch() {

    try {
      actionLabel.setText(" ");
      renderStartTime = 0;
      String pattern = patternField.getText();
      if (HTMLTools.isEmpty(pattern)) {
        Toolkit.getDefaultToolkit().beep();
        return;
      }

      patternField.commit();
      searchButton.setEnabled(false);
      EngineSet engineSet = createEngineSet();

      int sortOrder = alphabeticalButton.isSelected() ?
        Searcher.SORT_ALPHABETICALLY : Searcher.SORT_BY_RELEVANCE;

      int flags = 0;
      if (Config.getBoolean(FOREIGN_FILTER, FOREIGN_FILTER_DEFAULT))
        flags |= EngineRequest.FOREIGN_FILTER;
      if (Config.getBoolean(GARBAGE_FILTER, GARBAGE_FILTER_DEFAULT))
        flags |= EngineRequest.GARBAGE_FILTER;
       final EngineRequest req = new EngineRequest(pattern, 0,
        Config.getInt(MATCH_COUNT, MATCH_COUNT_DEFAULT));
      req.setFlags(flags);
      req.setTimeout(Config.getInt(SEARCH_TIMEOUT,
SEARCH_TIMEOUT_DEFAULT));
```

```
final Searcher searcher = new Searcher(engineSet, req, sortOrder);
        Runnable afterJob = new Runnable() {
        public void run() {
           ArrayList results = searcher.getResults();
           resultPane.setContentType("text/html");
           StringBuffer html = new StringBuffer();
           html.append("<HTML>\n");
           html.append("<HEAD></HEAD>\n");
           if (Config.isWindows())
              html.append("<BODY style=\"font-family: Verdana, Arial, Helvetica, sans-
serif;\">\n");
           else
           html.append("<BODY>\n");
           html.append("<FONT SIZE=-2>");
           html.append(searcher.getStats());
           html.append("</FONT>\n<BR>");

           resultPane.setText(html.toString()+"</BODY></HTML>");
           resultPane.setCaretPosition(0);

           if (results != null && results.size() > 0) {
              Iterator it = results.iterator();
              int index = 1;
              while(it.hasNext()) {
                 Hit hit = (Hit) it.next();
                 html.append(index).append(')').append(' ');
                 hit.pasteHTML(html);
                 html.append(BR);
                 index++;
              }
           } else if (!searcher.isInterrupted()) {
              html.append("<H1>Nothing found</H1>");
           }

           html.append("\n</BODY></HTML>");
           try {

              if (previousFile != null && previousFile.exists()) {
                 previousFile.delete();
                 previousFile = null;
              }
              File resultsFile = File.createTempFile("isearch", ".html");
              previousFile = resultsFile;
              resultsFile.deleteOnExit();
```

```
            FileWriter fw = new FileWriter(resultsFile);
            fw.write(html.toString());
            fw.close();
            startAnimation("Loading");
            renderStartTime = System.currentTimeMillis();
            resultPane.setPage(resultsFile.toURL());

          } catch(Exception ex) {
            ex.printStackTrace();
            startAnimation("Loading");
            resultPane.setText(html.toString());
            stopAnimation();
          }
          resultPane.setCaretPosition(0);
        }};

        ProgressWindow controller = new ProgressWindow(this, getTitle(),
          "Performing Search. Please Wait...",
          searcher, afterJob);
        controller.setVisible(true);

      } catch (Exception ex) {
        ex.printStackTrace();
        statusBar.setText(ex.getClass().getName());
        resultPane.setText("");

      } finally {
        searchButton.setEnabled(true);
        resultPane.setCaretPosition(0);
      }

    }

    private void pageChanged(PropertyChangeEvent evt) {
      if (debug) Debug.println("pageChanged: old="+evt.getNewValue()+";
new="+evt.getNewValue());

      if (previousFile != null && previousFile.exists()) {
        previousFile.delete();
        previousFile = null;
      }

      stopAnimation();
      actionLabel.setText("Ready");
```

```
    if (debug) {
      if (renderStartTime != 0) {
        long t = System.currentTimeMillis() - renderStartTime;
        Debug.println("HTML load took "+t+" ms");
      }
    }
  }

  private void startAnimation(String action) {
    actionLabel.setText(action);
  }

  private void stopAnimation() {
    actionLabel.setText(" ");
  }

  public void setVisible(boolean b) {
    super.setVisible(b);
    if (!b) {
      Config.save();
      System.exit(0);
    }
  }

  void resultPane_hyperlinkUpdate(HyperlinkEvent e) {
    if (e.getEventType() == HyperlinkEvent.EventType.ENTERED) {
      statusBar.setText( HTMLTools.nvl( e.getDescription() ) );
    } else if (e.getEventType() == HyperlinkEvent.EventType.EXITED) {
      statusBar.setText( " " );
    } else if (e.getEventType() == HyperlinkEvent.EventType.ACTIVATED) {
      String url = e.getDescription();
      if (!HTMLTools.isEmpty(url)) openURL(url);
    }

  }

  private static final RE browseRE = new UncheckedRE("%1");
   private void openURL(String url) {
     String command = "";
     try {
      boolean defaultBrowser = !Config.getBoolean(USE_CUSTOM_BROWSER,
USE_CUSTOM_BROWSER_DEFAULT);
       if (Config.isWindows() && defaultBrowser) {
          File shortcut = File.createTempFile("kse", ".url");
```

```
            try {
               FileWriter writer = new FileWriter(shortcut);
               writer.write("[InternetShortcut]\nURL="+url);
               writer.close();
               command = "rundll32 url.dll,FileProtocolHandler
"+shortcut.getAbsolutePath();

            } finally {
               shortcut.deleteOnExit();
            }

         } else {

            String pattern = Config.getString(BROWSE_COMMAND, "");
            if (Config.isEmpty(pattern)) {
               JOptionPane.showMessageDialog(this, "Browser command line is not
specified.\n"+
                  "Open configuration and specify browser command.",
                  "Error", JoptionPane.ERROR_MESSAGE);
               return;
            }
            String escapedURL;
            if (Config.isWindows()) {
               escapedURL = "\""+url+"\"";
            } else {
               escapedURL = url;
            }
            command = browseRE.substituteAll(pattern, escapedURL);
         }

         Process p = Runtime.getRuntime().exec(command);


      } catch (Throwable ex) {
         ex.printStackTrace();
         String msg = ex.getMessage();
         String text = "Couldn't launch browser:\nCommand Line: "+command;
         if (msg != null) text = text + "\nError: "+msg;
         text = text + "\n\nReconfigure browser support.";
         JoptionPane.showMessageDialog(this, text, "Error",
JoptionPane.ERROR_MESSAGE);
      }

   }
```

```
  void menuFileExit_actionPerformed(ActionEvent e) {
    System.exit(0);
  }
  void this_componentResized(ComponentEvent e) {
    if (isVisible()) {
      Config.setGeometry(FRAME_GEOMETRY, getBounds());
    }
  }
  void menuToolsConfiguration_actionPerformed(ActionEvent e) {
    ConfigWindow wnd = new ConfigWindow(this, "Configuration");
    wnd.setVisible(true);
    wnd.dispose();

  }
  protected void processKeyEvent(KeyEvent e) {
    int code = e.getKeyCode();
    if (code == KeyEvent.VK_ENTER) {
      performSearch();
    }
    super.processKeyEvent( e );
  }
  protected void processEvent(AWTEvent e) {
    super.processEvent( e);
  }
  void relevanceButton_actionPerformed(ActionEvent e) {
    Config.setBoolean(ALPHABETICAL_SORT, alphabeticalButton.isSelected());
  }
  void alphabeticalButton_actionPerformed(ActionEvent e) {
    Config.setBoolean(ALPHABETICAL_SORT, alphabeticalButton.isSelected());
  }
```

17) ProgressWindow.java

```
package kwokchan.isearch.gui;
import java.awt.BorderLayout;
import java.awt.Frame;
import java.awt.event.ActionEvent;
import java.awt.event.WindowEvent;
import javax.swing.*;
import javax.swing.border.Border;
import kwokchan.isearch.Searcher;

public final class ProgressWindow extends JDialog implements Runnable {
```

```
JPanel mainPanel = new JPanel();
  BorderLayout borderLayout1 = new BorderLayout();
  Searcher searcher;
  Runnable afterJob;
  JLabel caption = new JLabel();
  Border border1;
  Frame frame;
  JPanel buttonPanel = new JPanel();
  JButton cancelButton = new JButton();

  public ProgressWindow(Frame frame, String title, String message, Searcher searcher,
Runnable afterJob) {
    super(frame, title, false);
    this.frame = frame;
    this.searcher = searcher;
    this.afterJob = afterJob;
    try {
      jbInit();
      caption.setText(message);
      pack();
      setLocationRelativeTo(frame);
      setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
    }
    catch(Exception ex) {
      ex.printStackTrace();
    }
  }

  public void setVisible(boolean b) {
    if (b) {
      super.setVisible(b);
      Thread worker = new Thread(this);
      worker.start();
    } else {
      super.setVisible(b);
    }
  }
  void jbInit() throws Exception {
    border1 = BorderFactory.createEmptyBorder(20,20,20,20);
    mainPanel.setLayout(borderLayout1);
    caption.setHorizontalAlignment(SwingConstants.CENTER);
    caption.setText("Performing search. Please wait");
    mainPanel.setBorder(border1);
    cancelButton.setText("Cancel");
```

```
        cancelButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(ActionEvent e) {
                cancelButton_actionPerformed(e);
            }
        });

        this.addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                this_windowClosing(e);
            }
        });

        getContentPane().add(mainPanel);
        mainPanel.add(caption, BorderLayout.NORTH);
        mainPanel.add(buttonPanel,  BorderLayout.SOUTH);
        buttonPanel.add(cancelButton, null);
    }

    public void run() {
        searcher.run();
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                setVisible(false);
                if (afterJob != null) afterJob.run();
            }
        });
    }

    void cancelButton_actionPerformed(ActionEvent e) {
        searcher.interrupt();
    }

    void this_windowClosing(WindowEvent e) {
        if (searcher != null && !searcher.isCompleted() && !searcher.isInterrupted()) {
            return;
        } else {
            dispose();
        }
    }

}
```

18) AltavistaAdaptor.java

```
package kwokchan.isearch.engine;
import gnu.regexp.RE;
import gnu.regexp.REMatch;
import gnu.regexp.UncheckedRE;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.net.*;
import java.util.ArrayList;
import java.util.Hashtable;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import kwokchan.isearch.*;

public class AltavistaAdaptor implements EngineAdaptor {
   static boolean debug = false;
   private static java.io.PrintStream Debug = System.out;
   private Hashtable props;
   private static RE[] intraSiteRE = {
      new UncheckedRE("%3A%2F%2Fbabel.altavista.com")
   };

   private static RE hitRE = new UncheckedRE(
      "<a\\s+class='res'\\s+href\\s*=\\s*'[^']+\\*\\*(http[^']+)'[^>]*?>(.*?)</a>",
      RE.REG_ICASE|RE.REG_DOT_NEWLINE
   );

   public AltavistaAdaptor() {
      props = new Hashtable();
   }

   public String getID() {
      return "altavista";
   }

   public String getName() {
      return "Altavista";
   }

   public String getHTMLName() {
      return "<FONT COLOR=#025700>Altavista</FONT>";
   }
```

```
public String getEngineURL() {
  return "www.altavista.com";
}

public Icon getIcon(int iconType) {
  try {
    URL url = getClass().getClassLoader().getResource("altavista.gif");
    return new ImageIcon(url);
  } catch (Throwable ex) {
    return null;
  }
}

public String getProperty(String name) {
  return (String) props.get(name);
}

public ArrayList parse(String html, EngineRequest request) {
  try {
    long t1 = System.currentTimeMillis();
    String body = HTMLTools.removeComments(html);
    body = HTMLTools.removeScripts(body);

    REMatch matches[] = hitRE.getAllMatches(body);
    long t2 = System.currentTimeMillis();
    if (debug) {
      Debug.println("altavista parse: "+(t2-t1)+" ms");
      try {
        File f = new File("altavista_tmp.html");
        FileWriter fw = new FileWriter(f);
        fw.write(body);
        fw.close();
        Debug.println("altavista body dumped to "+f.getAbsolutePath());
      } catch (Exception e) {
      }
    }
    int disqualified = 0;
    if (matches != null) {
      ArrayList list = new ArrayList(matches.length);
      if (debug) Debug.println("altavista: "+matches.length+" matches");
      for(int i=0,n=matches.length; i<n; i++) {
        REMatch match = matches[i];
        String target = match.toString(1);
        String title  = match.toString(2);
        disqualified++;
```

```
            if (!targetQualifies(target)) continue;
            if (!titleQualifies(title)) continue;
            disqualified--;
            target = unmangle(target);
            title = HTMLTools.removeTags(title);
            String desc = getDescription(match, target, body);
            Hit hit = new Hit(this, target, title, desc);
            list.add(hit);
          }
          if (debug) Debug.println("altavista: disqualified "+disqualified+" matches");
          return list;
        }
      } catch (Exception ex) {
        ex.printStackTrace();
      }
      return null;
    }

    private String getDescription(REMatch match, String target, String body) {
      int afterMatch = match.getEndIndex();
      int upperLimit = body.indexOf("<br>", afterMatch);
      if (upperLimit != -1) upperLimit = body.indexOf("<br>", upperLimit+4);
      if (upperLimit == -1) return null;
      int i = body.indexOf("<a", afterMatch);
      if (i != -1 && i < upperLimit) upperLimit = i;
      i = body.indexOf(target, afterMatch);
      if (i != -1 && i < upperLimit) upperLimit = i;

      String block = body.substring(afterMatch, upperLimit).trim();
      block = HTMLTools.removeTags(block).trim();
      if (block.startsWith("URL:")) block = block.substring(4).trim();

      if (block.length() > 0) return block;
      else return null;
    }

    private void applyProperties(URLConnection connection) {
      connection.setRequestProperty("User-Agent","Lynx/2.8.2");
    }

    public String[] performRequest(EngineRequest request, MutableBoolean interuptor,
long deadline) throws IOException {
```

```
        StringBuffer b = new StringBuffer("http://www.altavista.com/web/results?");
        b.append("q=").append(URLEncoder.encode(request.getPattern()));
        if (request.getStartIndex() > 0)
        {
           b.append("&stq=").append(request.getStartIndex());
        }
        b.append("&nbq=").append(request.getMatchCount());
        b.append("&kgs=0&kls=0");

        URL url = new URL(b.toString());
        HttpURLConnection http = (HttpURLConnection) url.openConnection();
        applyProperties(http);

        if (interuptor.getValue()) throw new IOException("Search cancelled");
        String page = HTMLTools.readContent(http, deadline);

        return new String[] { page };

    }
    private String unmangle(String target) {
        try {
           target = URLDecoder.decode( target );
        } catch (Exception ex) {
           ex.printStackTrace();
        }
        return target;
    }

    private boolean targetQualifies(String targetURL) {
        if (debug) Debug.println("altavista target URL: |" + targetURL + "|");
        if (HTMLTools.isEmpty(targetURL)) return false;

        for(int i=0,n=intraSiteRE.length; i<n; i++) {
           RE re = intraSiteRE[i];
           if (re.getMatch(targetURL) != null) return false;
        }
        return true;
    }

    private boolean titleQualifies(String title) {
        return true;
    }

}
```

19) GoogleAdaptor.java

```
package kwokchan.isearch.engine;
import gnu.regexp.RE;
import gnu.regexp.REMatch;
import gnu.regexp.UncheckedRE;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.net.*;
import java.util.ArrayList;
import java.util.Hashtable;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import kwokchan.isearch.*;
public class GoogleAdaptor implements EngineAdaptor {
   static boolean debug = false;
   private static java.io.PrintStream Debug = System.out;
   private Hashtable props;
   private static RE[] intraRE = {
      new UncheckedRE("google\\.com/\\w+\\?q="),
      new UncheckedRE("\\?q=cache:"),
      new UncheckedRE("[\\?|\\&]q=related")
   };

   private static RE hyperLinkRE =
      new UncheckedRE("<a\\s+href\\s*=\"(.*?)\"\\s+class=l\\s*>(.*?)</a>",
RE.REG_ICASE|RE.REG_DOT_NEWLINE);

   public GoogleAdaptor() {
      props = new Hashtable();
   }

   public String getID() {
      return "google";
   }

   public String getName() {
      return "Google";
   }

   public String getHTMLName() {
      return "<FONT COLOR=red>Google</FONT>";
   }
```

```
public String getProperty(String name) {
   return (String) props.get(name);
}

public String getEngineURL() {
   return "www.google.com";
}

public Icon getIcon(int iconType) {
   try {
      URL url = getClass().getClassLoader().getResource("google_logo.gif");
      return new ImageIcon(url);
   } catch (Throwable ex) {
      return null;
   }
}
public ArrayList parse(String html, EngineRequest request) {
   try {
      long t1 = System.currentTimeMillis();
      String body = HTMLTools.removeComments(html);
      REMatch matches[] = hyperLinkRE.getAllMatches(body);
      long t2 = System.currentTimeMillis();
      if (debug) {
         Debug.println("google parse: "+(t2-t1)+" ms");
         try {
            File f = new File("google_tmp.html");
            FileWriter fw = new FileWriter(f);
            fw.write(body);
            fw.close();
            Debug.println("google body dumped to "+f.getAbsolutePath());
         } catch (Exception e) {
         }
      }
      int disqualified = 0;
         if (matches != null) {
         ArrayList list = new ArrayList(matches.length);
         if (debug) Debug.println("google parse: "+matches.length+" RE matches");
         for(int i=0,n=matches.length; i<n; i++) {
            REMatch match = matches[i];
            String target = match.toString(1);
            if (!qualifies(target)) {
               disqualified++;
               continue;
            }
```

```
                target = unmangle(target);
                String title  = HTMLTools.removeTags(match.toString(2));
                String desc = HTMLTools.removeTags( getDescription(match, target, body)
);
                Hit hit = new Hit(this, target, title, desc);
                list.add(hit);
              }
            if (debug) Debug.println("google parse: disqualified "+disqualified+" matches");
              return list;
            } else {
              if (debug) Debug.println("google parse: no <A> RE matches");
            }
          } catch (Exception ex) {
            ex.printStackTrace();
          }
          return null;
        }
        private static final RE descRE = new
UncheckedRE("Description:</font></span>(.*?)<br>", RE.REG_ICASE);
        private static final RE translateRE = new UncheckedRE("\\[\\s*<a
href=http://translate\\.google\\.com.*?</a>\\s*\\]", RE.REG_ICASE);
        private String getDescription(REMatch match, String target, String body) {
          int afterMatch = match.getEndIndex();
          int upperLimit = body.indexOf("<p>", afterMatch);
          if (upperLimit == -1) return null;
          String block = body.substring(afterMatch, upperLimit);
          block = translateRE.substitute(block, "");
          if (target.startsWith("http://")) target = target.substring(7);
          int i = block.indexOf(target);
          if (i == -1) return null;
          block = block.substring(0, i);
           i = block.indexOf("<a");
          if (i != -1) block = block.substring(0, i);
          i = block.indexOf("Category: </span>");
          if (i != -1) block = block.substring(0, i);
          REMatch explicitMatch = descRE.getMatch(block);
          if (explicitMatch != null) {
            block = explicitMatch.toString(1).trim();
          } else {
            block = HTMLTools.removeTags(block).trim();
            if (block.startsWith("...")) block = block.substring(4).trim();
          }
          if (block != null && block.length() == 0) return null;
```

```
    return block;
      }

  public URL createURL(EngineRequest req)  {
    StringBuffer q = new StringBuffer("http://www.google.com/search?q=");
    q.append(URLEncoder.encode(req.getPattern()));
    q.append("&hl=en");
    if (req.getStartIndex() > 0) q.append("&start=").append(req.getStartIndex());
    if (req.getMatchCount() > 0) q.append("&num=").append(req.getMatchCount());
    if (req.getStartIndex() > 0) q.append("&sa=N");
    try {
      return new URL(q.toString());
    } catch (MalformedURLException ex) {
      ex.printStackTrace();
      return null;
    }
  }

  private void applyProperties(URLConnection connection) {
    connection.setRequestProperty("User-Agent", "Lynx/2.8.2");
  }

  public String[] performRequest(EngineRequest request, MutableBoolean
cancelRequest, long deadline) throws IOException {
    URL url = createURL(request);
    if (debug) Debug.println("google url="+url.toString());
    HttpURLConnection http = (HttpURLConnection) url.openConnection();
    applyProperties(http);
    String content = HTMLTools.readContent(http, deadline);
    return new String[] { content };

  }

  private static RE mangledRE = new UncheckedRE("q=(http://[^&\\s\">]+)",
RE.REG_ICASE);
   private String unmangle(String url) {
    if (url.startsWith("/url?")) {
      REMatch match = mangledRE.getMatch(url);
      if (match != null) {
        return match.toString(1);
      }
    }
    return url;
  }
```

```
    private boolean qualifies(String targetURL) {
    if (debug) Debug.println("google target URL: |" + targetURL + "|");
    if (HTMLTools.isEmpty(targetURL)) return false;
    if (targetURL.startsWith("/url?sa=U") && targetURL.indexOf("q=http") != -1)
return true;
    if (!targetURL.startsWith("http://")) return false;
    if (targetURL.startsWith("http://directory.google.com/")) return false;
    if (targetURL.startsWith("http://translate.google.com/")) return false;
    for(int i=0,n=intraRE.length; i<n; i++) {
      if (intraRE[i].getMatch(targetURL) != null) return false;
    }
    return true;
  }
}
```

20) HotbotAdaptor.java

```
package kwokchan.isearch.engine;
import gnu.regexp.RE;
import gnu.regexp.REMatch;
import gnu.regexp.UncheckedRE;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.net.*;
import java.util.ArrayList;
import java.util.Hashtable;
import java.util.Vector;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import kwokchan.isearch.*;
public class HotbotAdaptor implements EngineAdaptor {
  static boolean debug = false;
  private static java.io.PrintStream Debug = System.out;
  private Hashtable props;
  private static final int HOTBOT_PAGE_SIZE = 10;
  private static RE[] intraLycosRE = {
    new UncheckedRE("://(help|search|www|dir|hotbot)\\.lycos\\.com/",
RE.REG_ICASE),
    new UncheckedRE("://r\\.hotbot\\.com/", RE.REG_ICASE),
    new UncheckedRE("://click\\.hotbot\\.com/director\\.asp\\?.*?id=[^\\d]",
RE.REG_ICASE),
    new UncheckedRE("(&|&amp;)position=\\d+", RE.REG_ICASE)
  };
```

```
  private static RE[] intraLycosTitleRE = {
    new UncheckedRE("<IMG", RE.REG_ICASE)
  };
  private static RE hotbotRE = new UncheckedRE(
    "target=(http%3A[^&]+)"
  );
  private static RE hitRE = new UncheckedRE(
      "<a[^>]+href\\s*=\"([^\"]+)\"[^>]*>([^\n\r]*?)</a>",
    RE.REG_ICASE|RE.REG_DOT_NEWLINE
  );
  private static RE ampRE = new UncheckedRE("&amp;", RE.REG_ICASE);

  private static RE nextPageRE = new UncheckedRE(
    "<a href=\"(default\\.asp\\?[^\">]*first=[^\">]+page=more[^\"]*)\"",
RE.REG_ICASE
  );

  public HotbotAdaptor() {
    props = new Hashtable();
  }

  public String getID() {
    return "hotbot";
  }

  public String getName() {
    return "HotBot";
  }

  public String getHTMLName() {
    return "<FONT COLOR=#570200>HotBot</FONT>";
  }

  public String getEngineURL() {
    return "www.hotbot.com";
  }

  public Icon getIcon(int iconType) {
    try {
      URL url = getClass().getClassLoader().getResource("hotbot.gif");
      return new ImageIcon(url);
    } catch (Throwable ex) {
      return null;
    }
  }
```

```
public String getProperty(String name) {
   return (String) props.get(name);
}
public ArrayList parse(String html, EngineRequest request) {
   try {
      long t1 = System.currentTimeMillis();
      String body = HTMLTools.removeComments(html);
      body = HTMLTools.removeScripts(body);
      REMatch matches[] = hitRE.getAllMatches(body);
      long t2 = System.currentTimeMillis();
      if (debug) Debug.println("hotbot parse: "+(t2-t1)+" ms");
      if (matches != null) {
         ArrayList list = new ArrayList(matches.length);
         for(int i=0,n=matches.length; i<n; i++) {
            REMatch match = matches[i];
            String target = match.toString(1);
            String title  = match.toString(2);
            if (!targetQualifies(target)) continue;
            if (!titleQualifies(title)) continue;
            target = unmangle(target);
            title = HTMLTools.removeTags(title);
            String desc = getDescription(match, target, body);
            Hit hit = new Hit(this, target, title, desc);
            list.add(hit);
         }
         return list;
      }
   } catch (Exception ex) {
      ex.printStackTrace();
   } return null;
}
private String getDescription(REMatch match, String target, String body) {
   int afterMatch = match.getEndIndex();
   int upperLimit = body.indexOf("<br>", afterMatch);
   if (upperLimit != -1) upperLimit = body.indexOf("<br>", upperLimit+4);
   if (upperLimit == -1) return null;
   int i = body.indexOf("<a", afterMatch);
   if (i != -1 && i < upperLimit) upperLimit = i;
   i = body.indexOf(target, afterMatch);
   if (i != -1 && i < upperLimit) upperLimit = i;
   String block = body.substring(afterMatch, upperLimit);
   block = HTMLTools.removeTags(block).trim();
   if (block.length() > 0) return block;
   else return null;
}
```

```
  private void applyProperties(URLConnection connection)
   {
    connection.setRequestProperty("User-Agent", "Mozilla/4.0 (compatible; MSIE 6.0;
.NET CLR 1.1.4322)");
      connection.setRequestProperty("Accept-Language", "en-us");
      connection.setRequestProperty("Accept", "*/*");
      connection.setRequestProperty("Cookie", "prov=hotbot");

   }

   public String[] performRequest(EngineRequest request, MutableBoolean interuptor,
long deadline) throws IOException {

      StringBuffer b = new StringBuffer("http://www.hotbot.com/default.asp?query=");
      b.append(URLEncoder.encode(request.getPattern()));
      b.append("&ps=");
      if (request.getStartIndex() > 0)
      {
        b.append("&first=").append(request.getStartIndex());
        b.append("&page=more&ca=w");
      }
      else
      {
        b.append("&loc=searchbox&tab=web");
      }
      b.append("&provKey=Inktomi&prov=HotBot");

      URL url = new URL(b.toString());
      HttpURLConnection http = (HttpURLConnection) url.openConnection();
      applyProperties(http);

      if (interuptor.getValue()) throw new IOException("Search cancelled");
      String page = HTMLTools.readContent(http, deadline);
      Vector pages = new Vector();
      int hotbotPagesRead = 1;
      int lastRequestedHit = request.getStartIndex() + request.getMatchCount();
      pages.add(page);
      URL referer = url;

      while(true) {
        if (interuptor.getValue()) throw new IOException("Search cancelled");
        if (deadline > 0 && System.currentTimeMillis() > deadline) break;
        if ((hotbotPagesRead * HOTBOT_PAGE_SIZE) >= lastRequestedHit) break;
        REMatch nextURLMatch = nextPageRE.getMatch(page);
```

```
        if (nextURLMatch == null) break;
        String nextURLText = nextURLMatch.toString(1);
        if (!nextURLText.startsWith("http://"))
        nextURLText = "http://www.hotbot.com/"+nextURLText;
        nextURLText = ampRE.substituteAll(nextURLText, "&");
        if (debug) Debug.println("nextURL="+nextURLText);
        URL nextURL = new URL(nextURLText);
        http = (HttpURLConnection) nextURL.openConnection();
        applyProperties(http);
        http.setRequestProperty("Referer", referer.toString());
        referer = nextURL;
        page = HTMLTools.readContent(http, deadline);
        pages.add(page);
        hotbotPagesRead++;
      }

      if (debug) {
        try {
          File f = new File("hotbot_tmp.html");
          FileWriter fw = new FileWriter(f);
          fw.write(page);
          fw.close();
          Debug.println("hotbot body dumped to "+f.getAbsolutePath());
        } catch (Exception e) {
        }
      }

      return (String[]) pages.toArray(new String[pages.size()]);

    }

    private String unmangle(String target) {
      REMatch match = hotbotRE.getMatch(target);
      if (match != null) {
        target = URLDecoder.decode( match.toString(1) );
      }
      return target;
    }

    private boolean targetQualifies(String targetURL) {
      if (debug) Debug.println("hotbot target URL: |" + targetURL + "|");
      if (HTMLTools.isEmpty(targetURL)) return false;
      if (!targetURL.startsWith("http://")) return false;
      for(int i=0,n=intraLycosRE.length; i<n; i++) {
        RE re = intraLycosRE[i];
```

```
            if (re.getMatch(targetURL) != null) return false;
        }
        return true;
    }

    private boolean titleQualifies(String title) {
        for(int i=0,n=intraLycosTitleRE.length; i<n; i++) {
            RE re = intraLycosTitleRE[i];
            if (re.getMatch(title) != null) return false;
        }
        return true;
    }
}
```

21) LycosAdaptor.java

```
package kwokchan.isearch.engine;
import gnu.regexp.RE;
import gnu.regexp.REMatch;
import gnu.regexp.UncheckedRE;
import java.io.IOException;
import java.net.*;
import java.util.ArrayList;
import java.util.Hashtable;
import java.util.Vector;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import kwokchan.isearch.*;

public class LycosAdaptor implements EngineAdaptor {

    static boolean debug = false;
    private static java.io.PrintStream Debug = System.out;
    private static final int LYCOS_PAGE_SIZE = 10;
    private Hashtable props;
    private static RE nextPageRE = new UncheckedRE(
        "<a href=\"(default\\.asp\\?[^\">]*first=[^\">]+)\"[^>]*>.*Next.*</a>",
RE.REG_ICASE
    );

        private static RE[] intraLycosRE = {
        new UncheckedRE("://(help|search|www|dir|hotbot)\\.lycos\\.com/",
RE.REG_ICASE),
```

```
      new UncheckedRE("://r\\.(hotbot)\\.com/", RE.REG_ICASE),
      new UncheckedRE("://click\\.(hotbot|lycos)\\.com/director\\.asp\\?.*?id=[^\\d]",
RE.REG_ICASE),
      new UncheckedRE("(&|&amp;)position=\\d+", RE.REG_ICASE)
    };

    private static RE[] intraLycosTitleRE = {
      new UncheckedRE("<IMG", RE.REG_ICASE)
    };

    private static RE mangledRE = new UncheckedRE(
      "target=(http%3A[^&]+)"
    );

    private static RE hitRE = new UncheckedRE(
      "<a\\s+href\\s*=\"([^\"]+)\"\\s*[^>]*>([^\r\n]+)</a>",
      RE.REG_ICASE|RE.REG_DOT_NEWLINE
    );

    private static RE ampRE = new UncheckedRE("&amp;", RE.REG_ICASE);
    public LycosAdaptor() {
      props = new Hashtable();
    }

    public String getID() {
      return "lycos";
    }

    public String getName() {
      return "Lycos";
    }

    public String getHTMLName() {
      return "<FONT COLOR=green>Lycos</FONT>";
    }

    public String getEngineURL() {
      return "search.lycos.com";
    }

    public Icon getIcon(int iconType) {
      try {
        URL url = getClass().getClassLoader().getResource("small_lycos.gif");
        return new ImageIcon(url);
```

```
    } catch (Throwable ex) {
      return null;
    }
}

public String getProperty(String name) {
    return (String) props.get(name);
}

public ArrayList parse(String html, EngineRequest request) {
    try {
      long t1 = System.currentTimeMillis();
      String body = HTMLTools.removeComments(html);
      body = HTMLTools.removeScripts(body);
       REMatch matches[] = hitRE.getAllMatches(body);
        long t2 = System.currentTimeMillis();
      if (debug) Debug.println("lycos parse: "+(t2-t1)+" ms");
      if (matches != null) {
        ArrayList list = new ArrayList(matches.length);
        for(int i=0,n=matches.length; i<n; i++) {
          REMatch match = matches[i];
          String target = match.toString(1);
          String title  = match.toString(2);
          if (!targetQualifies(target)) continue;
          if (!titleQualifies(title)) continue;
          target = unmangle(target);
          title = HTMLTools.removeTags(title);
          String desc = getDescription(match, target, body);
          Hit hit = new Hit(this, target, title, desc);
          list.add(hit);
          if (list.size() >= request.getMatchCount()) break;
        }
        return list;
      }
    } catch (Exception ex) {
      ex.printStackTrace();
    }
    return null;
}

private String getDescription(REMatch match, String target, String body) {
    int afterMatch = match.getEndIndex();
    int upperLimit = body.indexOf("<br>", afterMatch);
    if (upperLimit == -1) return null;
```

```
      int i = body.indexOf("<a", afterMatch);
      if (i != -1 && i < upperLimit) upperLimit = i;
      i = body.indexOf(target, afterMatch);
      if (i != -1 && i < upperLimit) upperLimit = i;
      String block = body.substring(afterMatch, upperLimit);
      block = HTMLTools.removeTags(block).trim();
      if (block.startsWith("-")) block = block.substring(1).trim();
      if (block.length() > 0) return block;
      else return null;
    }

    private void applyProperties(URLConnection connection) {
      connection.setRequestProperty("User-Agent", "Mozilla/4.0 (compatible; MSIE 6.0;
.NET CLR 1.1.4322)");
    }

    public String[] performRequest(EngineRequest request, MutableBoolean interuptor,
long deadline) throws IOException {
      String initialRequest =
"http://search.lycos.com/default.asp?loc=searchbox&tab=web&query="+URLEncoder.en
code(request.getPattern());
      URL url = new URL(initialRequest);
      HttpURLConnection http = (HttpURLConnection) url.openConnection();
      applyProperties(http);
      Vector pages = new Vector();
      String page = HTMLTools.readContent(http, deadline);
      int lycosPagesRead = 1;
      int lastRequestedHit = request.getStartIndex() + request.getMatchCount();
      pages.add(page);

      URL referer = url;
      while(true) {
        if (interuptor.getValue()) throw new IOException("Search cancelled");
        if (deadline > 0 && System.currentTimeMillis() > deadline) break;
        if ((lycosPagesRead * LYCOS_PAGE_SIZE) >= lastRequestedHit) break;
        REMatch nextURLMatch = nextPageRE.getMatch(page);
        if (nextURLMatch == null) break;
        String nextURLText = nextURLMatch.toString(1);
        if (!nextURLText.startsWith("http://"))
          nextURLText = "http://search.lycos.com/"+nextURLText;
        nextURLText = ampRE.substituteAll(nextURLText, "&");
        if (debug) Debug.println("nextURL="+nextURLText);
        URL nextURL = new URL(nextURLText);
        http = (HttpURLConnection) nextURL.openConnection();
```

```
            applyProperties(http);
            http.setRequestProperty("Referer", referer.toString());
            referer = nextURL;
            page = HTMLTools.readContent(http, deadline);
            pages.add(page);
            lycosPagesRead++;
        }

        return (String[]) pages.toArray(new String[pages.size()]);
    }

    private String unmangle(String target) {
        REMatch match = mangledRE.getMatch(target);
        if (match != null) {
            try {
                target = URLDecoder.decode( match.toString(1) );
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }
        return target;
    }

    private boolean targetQualifies(String targetURL) {
        if (debug) Debug.println("lycos target URL: |" + targetURL + "|");
        if (HTMLTools.isEmpty(targetURL)) return false;
        if (!targetURL.startsWith("http://")) return false;
        for(int i=0,n=intraLycosRE.length; i<n; i++) {
            RE re = intraLycosRE[i];
            if (re.getMatch(targetURL) != null) return false;
        }
        return true;
    }

    private boolean titleQualifies(String title) {
        for(int i=0,n=intraLycosTitleRE.length; i<n; i++) {
            RE re = intraLycosTitleRE[i];
            if (re.getMatch(title) != null) return false;
        }
        return true;
    }
}
```