The Law of One Price in International Trade
Textbook Pricing Information from Online Bookstores

By

Choon Hong Tay

Department of Electrical and Computer Engineering, Carnegie Institute of Technology
Carnegie Mellon University

Submitted to the Department of Economics, College of Humanities and Social Sciences
Carnegie Mellon University
In fulfillment of the requirements for the Senior Honors Program

Advisor

Professor Karen Clay

Heinz School of Public Policy and Management,
Carnegie Mellon University

May 2001

Abstract

The Law of One Price (LOP) has a very simple premise: that when prices are converted to a common currency, the same product should sell for the same price anywhere in the world. In recent years, the rise of the Internet as an electronic commerce medium led many to believe that global markets are becoming more perfect and therefore prices should be converging. We collected weekly prices of 101 textbooks in 6 broad categories for 8 weeks. Automated price collection agents were sent to 5 online stores in North America and 4 in Western Europe. We found unit prices of books differed significantly between the two geographical regions. European booksellers price an average textbook at 71.8% of the price quoted by North American retailers. The margin of savings for the hypothetical US student depended on how long the student was able to wait. When standard shipping fees are added, European textbooks cost 79.6% of the same book obtained in North America. If we equalized the waiting time by choosing expedited shipping for European stores and standard shipping for North American stores, the average book costs the same.

Acknowledgements

I would like to thank my advisor, Professor Karen Clay for her support, guidance and understanding during the course of this project. Many thanks also to Kartik Hosanager who provided me with a lot of help in the technical aspects. Finally, to all my friends and family who believed in and supported me throughout the years, a heartfelt thanks.

Introduction

The Law of One Price (LOP) is one of the most frequently quoted economic laws but also one of the most commonly violated, the term "law" notwithstanding. The premise of LOP is very simple: that when prices are converted to a common currency, the same good should sell for the same price anywhere in the world. This concept is very appealing intuitively as its violation leads to market arbitrage opportunities. The existence of arbitrage should force international market prices to converge.

In the international market for gold, spot prices move closely in tandem across the world. Its rapid price convergence can be attributed to its historical role as a medium of monetary exchange. On the other hand, the famous "Big Mac Index"[1] shows that the price of a Big Mac is not the same everywhere McDonald's sells it.

	Big Mac Prices	
	Local Currency	US Dollars
Australia	A$2.59	$1.54
Britain	£1.90	$3.00
Canada	C$2.85	$1.94
France	FFr18.50	$2.62
Germany	DM4.99	$2.37
Hong Kong	HK$10.20	$1.31
Japan	¥294	$2.78
New Zealand	NZ$3.40	$1.69
Singapore	S$3.20	$1.88
Switzerland	SFr5.90	$3.48
United States	$2.51	$2.51

Table 1: The Big Mac Index

[1] http://economist.com/markets/Bigmac/Index.cfm

A major requirement for the LOP to work is that the markets have to be perfect. A. Marshal[2] believed that "the more nearly perfect a market is, the stronger the tendency for the same price to be paid for the same thing at the same time in all parts of the market". Referring back to the gold example, the market in which it is traded is essentially perfect. Regardless of the market, gold is virtually indistinguishable in terms of quality. However, Big Macs do not have the benefit of a perfect market. Labor and other costs vary substantially among countries, leading to dispersed prices even if the raw ingredients that go into a Big Mac are sourced from the same origin.

Search Costs and the Internet

Conventional economic theory predicts that whenever producers are placed in competition, prices will be driven down to the lowest cost levels and all producers will earn zero economic profit. This is in fact the Bertrand model of competition. Historically, technological advances played a major role in increasing the efficiency of markets. Markets are in competition only if consumers have access to all of them. Physically, the invention of steam, internal combustion and jet engines led to the introduction of trains, automobiles and airplanes respectively, which in turn introduced increasingly efficient channels of transportation, leading to markets being drawn into competition where they were relatively isolated entities before. The speed and volume of communications between markets also improved. The telegraph first introduced coordination at a distance; later, the telephone and facsimile dramatically increased the volume of trade between geographically disparate markets with its lower usage cost and wider reach. In advertising, radio and television made mass marketing possible, resulting in increased consumer information.

[2] Principles of Economics, London, 1890, p 325.

The introduction of the Internet brought with it improvements and new possibilities in commerce. Already, it has demonstrated many capabilities. For example, it is possible to purchase and download books, music and software, using the Internet as an instant physical delivery medium. The Internet also lowered communication costs through electronic mail, Internet phone or video-conferencing across international boundaries. It is also a popular medium for advertising, given its global reach and potential for consumer-targeted marketing.

These examples exemplify the use of the Internet to improve the reach and efficiency as well as lowering transaction costs for businesses. On the consumer side, the Internet promises more "perfect" access to information. Price comparison engines like Dealtime, Pricescan and MySimon are increasing their product and store coverage constantly, leading to reduced search costs. For a certain period of time, Dealtime gathered prices for textbooks from all over the world, which hinted that prices were not necessarily converged and which motivated this project. Interestingly, their search engines now only return prices for US-based retailers even if cheaper prices are available elsewhere.

Other Efforts

Past research in this area provided some understanding about the evolving nature of the Internet as an electronic commerce medium while also hinting toward what one might observe in the future. In the early years of the Internet, relatively lower penetration rates meant that any business conducted online was not viable. Advancements in technology lowered the costs of establishing an online business. Concurrently, access

costs borne by consumers decreased while speeds increased steadily, making shopping online a feasible and satisfying experience.

In an early work, Lee[3] found that auction prices for used cars in Japan was substantially higher online than in traditional markets. He postulated that one of the reasons include the fact that the online market consists of newer used cars, translating into higher prices. Bailey[4] in his 1998 Ph.D. thesis found that prices of books, compact discs and software during 1996-1997 were similarly higher on the Internet than in conventional retail channels.

Shortly after this early time period, technology advancement reached a critical threshold whereby e-businesses became viable. Companies aggressively established web presences and graduated quickly into full-fledged e-businesses. Amazon.com was created in 1995 and rapidly became one of the largest online booksellers. Barnes and Noble as well as Borders followed suit later, in 1997 and 1998 respectively, establishing their e-business on top of their brick and mortar shops. In a short period of two years, the online commerce landscape changed. Brynjolfsson and Smith in their study of book and compact disc prices for the period 1998-1999 found that prices on the Internet were now lower.

All of the prior work suggests that some form of adjustment and adaptation is occurring in response to the introduction of this new medium for commerce. Subsequently, the focus shifted to comparing prices among online retailers only. Clay,

[3] H.G. Lee, "Do Electronic Marketplaces Lower the Price of Goods?", Communications of the ACM, Vol. 41 No. 1, January 1998
[4] J.P. Bailey, "Intermediation and Electronic Markets: Aggregation and Pricing in Internet Commerce", PhD thesis, MIT, May 1998

Krishnan and Wolff[5] tracked the prices of more than 300 books in 3 different categories from August 1999 to January 2000. The data from 32 online bookstores in the United States suggest that prices did not converge to cost, implying that firms were actively differentiating their products so as not to compete on price.

In an earlier paper by the same authors, prices were found to be substantially different. Interestingly, however, price convergence was observed for the 3 biggest US retailers (Amazon, Barnesandnoble, Borders) in the subsequent work. This suggests that prices were converging slowly in the nascent U.S.-based Internet market.

Finally, this paper departs from the rest of the prior work with its international focus. A popular prediction is that the Internet will enable the creation of a global, borderless marketplace. It is therefore interesting to see if the virtual removal of geographical boundaries has already created a "perfect" marketplace.

Methodology

In this paper, we will define our market as consisting of online textbook retailers. The consumer is situated in the state of Pennsylvania, United States to simplify analysis. Ideally, a consumer should be placed in each of the countries surveyed to provide a symmetric viewpoint. However, our results regarding price convergence are not affected in any way without this mirror treatment.

Taxes, Customs Duties and the Cost of Waiting

In trying to make the product as homogeneous as possible, there are several complicating factors to take care of, including the problem of varying tax rates, customs duties and differences in waiting times.

[5] K. Clay, R. Krishnan and E. Wolff, "Prices and Price Dispersion on the Web: Evidence from the Online Book Industry", March 2001. http://www.heinz.cmu.edu/~kclay/bookpricing.pdf

The United Kingdom exempts books from VAT (Value Added Tax) while Germany imposes a 7% VAT automatically, which cannot be refunded in the case of textbooks. Canadian stores do not levy the GST (Goods and Services Tax) on international shipments. Stores in the United States levy the tax only if the store has a physical presence in the state. The customer is responsible for the payment of this "use tax", but it is seldom paid and the states do not enforce it strictly. As none of the U.S. based stores have a physical presence in the state of Pennsylvania, no sales tax was included.

United States customs allows each person a daily $200 duty-free limit on incoming international shipments. Since most books are not valued at more than $150, it is assumed that the consumer will not be levied any customs duty.

International shipments take varying times to arrive in the United States, where the average consumer in our analysis is located. In some instances, the opportunity cost of waiting for the shipment to arrive is negligible for a patient consumer. Thus, the longer waiting time associated with purchasing books from European booksellers does not matter. On the other hand, some consumers have more time-critical needs for the books. In this case, we used the additional cost of expedited shipping as a proxy for the increased waiting time.

On average, European books take an extra week to reach the consumer in the United States by standard shipping than with expedited shipping. Expedited shipping allows the consumer to receive the books in about the same time frame as a book shipped by standard services in the United States. However, expedited shipping is not an option at all the stores surveyed. Hence, we added the average expedited shipping fees quoted by

the European booksellers onto the unit prices to account for the difference in waiting time in a separate analysis.

Bookstore search

To evaluate the viability of the project, an Internet search for textbook retailers was conducted. Initial stores included:

Country	Stores
United States	Amazon, Barnesandnoble, Bigwords, Efollett, Ecampus, Varsitybooks
United Kingdom	Bol, Alphabetstreet, Studentbookworld, Amazon
Canada	Indigo, Chapters
Australia	Dymocks, Dadirect
New Zealand	Flyingpig
India	Fabmart
Singapore	Acmabooks, Mphonline
Germany	Amazon

Table 2: Available Online Textbook Retailers

Because the target consumer is based in the United States, located stores in Taiwan, Hong Kong, China, Japan, France and Austria were rejected as they mainly carried books in their own native languages.

In order to produce results that reflect intra-country competition, several stores were dropped in countries where there was only one store. The final list is shown below, grouped by geographical location:

Region	Stores
Domestic (US/CA)	Amazon (US), Barnesandnoble, Varsitybooks, Indigo, Chapters
Non-Domestic (UK/DE)	Amazon (UK), Alphabetstreet, Studentbookworld, Amazon (DE)

Table 3: Final Store Selections

With this categorization, shipping times can also be better accounted for, i.e. European shipments take on average an additional week to arrive in the US by regular mail service while shipments from other parts of world will need different varying times.

Textbook ISBN and Pricing Data Collection

A major feature of books is the International Standard Book Number (ISBN) assigned to all of them. The ISBN is unique and guarantees that the books in our sample are the same regardless of where it is purchased. In addition, the ISBN is frequently used directly by stores in their catalogs, leading to easier and more accurate identification of the books that we are tracking. As different bundles of books are issued different ISBNs, we are not concerned with different bundling options (e.g. additional CD-ROM discs or exercise supplements).

After the set of stores was decided, the next step was to gather a list of textbooks, preferably available in all of the stores to be tracked. Initially, we desired to gather textbook "bestsellers" in the likes of the books on the New York Times bestsellers' lists. Unfortunately, such lists are not available. A similar list was found in Amazon (US), but that list was found to be too general and tended not to list undergraduate textbooks.

In the end, random textbook titles and ISBNs were collected from Varsitybooks' excellent categorical listings. Textbooks in a wide spectrum of categories were chosen: Economics, Mathematics, Architecture, Humanities and Social Sciences, Engineering, Computer Science and Natural Sciences. In order to more accurately select textbooks from other general books that Varsitybooks carry, the following criteria were applied: title copyrighted on or after 1997, price greater than $60 and edition higher than first. The edition criterion was used as a proxy for identifying books in demand.

Automated agents were constructed using existing JAVA code from a related project and modified for the individual stores in our sample. Web pages with the unit price information were downloaded and parsed. Shipping costs and dispatch times were noted from each of the stores. Shipping costs were monitored periodically and were found to be unvarying over the period during which prices were collected. Due to many factors such as slow server response or server downtimes, prices were collected three times a week. The lowest price of the week was selected as the price for that week. In most cases, prices do not change or do so in very small amounts, so we do not expect any problems with temporal aggregation. Each day that prices were collected, the prevailing currency exchange rates were noted. The weekly average currency rate was used in the conversion to United States Dollars since customers generally do not know when their credit cards will be charged. Further, fluctuations in the currency exchange rates were relatively small in magnitude.

Prices for 101 books from the 9 online bookstores were therefore collected over an 8-week period spanning the period February to March 2001.

Predictions

There are 4 underlying assumptions about the Bertrand Model in order for the LOP to hold. These are: a) consumers are perfectly informed about firms' prices, b) goods are identical, c) firms are able to choose their own prices, and d) firms can supply whatever quantity the market demands at constant marginal cost.

While consumers were able to purchase goods from other countries in the past, this was usually a complicated affair. The consumer must be able to access the catalog of the foreign retailer, be it a mail-order catalog or an advertisement from a foreign

newspaper or magazine. This necessarily meant that the reach of foreign retailers in the domestic market was extremely limited to only the most informed consumers. If catalogs were not available, the consumer must call up the foreign retailer to ask about product availability and price, which increased the search costs. Therefore, markets were effectively separated geographically.

With the Internet, one can assume that the availability of search engines reduces the search costs while increasing the reach, i.e. consumers can find the prices of any product from any web-based retailer anywhere in the world. Unique identifiers like ISBNs provide us with the guarantee that the goods are identical.

Should the assumptions be true for the online market, we expect that prices will be the same for the same book regardless of which retailer it was purchased from.

Analysis and Results

The summary statistics for each store is produced in Table 4.

All prices in USD	N	Unit Price	Standard Deviation	Price Range	Normalized Price
Individual Stores					
Amazon (US)	797	91.44	16.06	50.67-120.95	1.01
Barnesandnoble (US)	808	88.08	16.71	49.95-120.95	0.98
Varsitybooks (US)	808	92.70	16.30	57.00-127.50	1.03
Chapters (CA)	688	72.61	14.58	36.94-124.12	0.81
Indigo (CA)	756	75.59	17.82	38.21-118.85	0.84
Alphabetstreet (UK)	400	44.05	13.01	20.25-124.01	0.49
Studentbookworld (UK)	588	54.47	26.59	20.13-152.79	0.61
Amazon (UK)	702	59.32	27.55	21.32-129.07	0.66
Amazon (DE)	701	77.36	28.91	22.73-130.65	0.86
Geographical Segregation					
Domestic (US/CA)	3857	84.58	18.30	36.94-127.50	0.94
Non-Domestic (UK/DE)	2391	60.86	28.41	20.13-152.79	0.68
United States (US)	2413	90.80	16.46	49.95-127.50	1.01
Canada (CA)	1444	74.17	16.42	36.94-124.12	0.83
United Kingdom (UK)	1690	54.02	25.22	20.13-152.79	0.60
Germany (DE)	701	77.36	28.91	22.73-130.65	0.86
Book Categories					
Economics	615	81.13	24.62	30.19-129.07	0.89
Mathematics	585	78.45	25.50	33.22-124.02	0.83
Sciences	508	88.57	29.67	33.67-130.32	0.84
Computer Science	639	64.45	17.28	34.17-118.80	0.87
Engineering	2367	83.30	23.98	32.79-152.79	0.80
Humanities	1534	60.38	20.07	20.13-104.03	0.87

Table 4: Summary Statistics

Our sample comprises 101 random books in 6 categories. Book availability ranges from 100% for Varsitybooks and BarnesandNoble to a low of 49.5% for Alphabetstreet. Average availability is 85.9%.

To aid comparison, we include normalized prices in Table 4. Normalized prices are obtained from dividing the unit prices by the publisher's recommended price in the United States using the Books in Print database. Recommended prices in the other

countries could not be obtained, hence this figure is purely a consumer-oriented measure, i.e. it measures the amount of savings the target consumer obtains rather than the amount of discount provided by the retailer.

The broad Engineering and Humanities categories comprise the majority of the books due to the many different specializations available. Engineering textbooks have the highest discounts while economics textbooks have the least. Nevertheless, the variation is only within 9% of the recommended price. This suggests that different types of books do not have different levels of discounts.

The results shown in Table 4 already show us that substantial price differences do exist. Overall, books sold in the United States market have the highest normalized unit prices, followed by Germany, Canada and the United Kingdom. Focusing on the stores, Varsitybooks not only do not discount their books but prices them 3% higher than the recommendation, while Alphabetstreet provides the most discount.

A series of regressions of prices versus market were performed and results are presented in the following tables.

Variable	Coefficient	Robust Standard Error	t-Value	Pr > |t|	R^2
Dependent Variable: Unit Price (N = 6247)					
UK	-36.7202	0.6781	-54.1501	<0.0001	0.3676
DE	-13.2874	1.1252	-11.8089	<0.0001	
CA	-16.5377	0.5327	-31.0454	<0.0001	
Time	0.0175	0.1127	0.1555	0.8764	
Domestic	23.6534	0.6392	37.0074	<0.0001	0.2395
Dependent Variable: Normalized Unit Price					
UK	-0.4057	0.0074	-55.0696	<0.0001	0.3833
DE	-0.1446	0.0113	-12.7851	<0.0001	
CA	-0.1773	0.0047	-37.8851	<0.0001	
Time	0.0007	0.0011	0.6015	0.5476	
Domestic	0.2627	0.0069	38.2588	<0.0001	0.2385

Table 5: Regression of Unit Prices (Controlling for Book Categories)

Table 5 shows the results of regressing unit prices against the dummy variable Domestic as well as the region specific dummies UK, DE and CA. White's test of heteroskedasticity shows that the data is indeed heteroskedastic. Therefore, robust standard errors are used in these and all of the following regressions. Our results verify that there is strong evidence of higher unit prices in the domestic market (defined to be United States and Canada). Going on to a more region specific viewpoint, unit prices are lower in the United Kingdom, Germany and Canada than in the United States. Regardless of the market, we find that prices do not change with time. This is expected as the retailers were observed to be keeping unit prices constant most of the time while currency rates were relatively stable as well.

The unit price is only one component of the total cost of a book. Other costs include shipping charges, taxes, customs duties and the cost of waiting. However, as mentioned before, we will not be charged any additional taxes nor customs duties given

our location and commodity. Thus, we are only interested in adding shipping charges and waiting time.

Variable	Coefficient	Robust Standard Error	t-Value	Pr > |t|	R^2
Dependent Variable: Total Price (N = 6247)					
UK	-33.4094	0.6854	-48.7415	<0.0001	0.3439
DE	-1.6011	1.1256	-1.4224	0.1550	
CA	-16.3733	0.5352	-30.5907	<0.0001	
Time	0.0162	0.1134	0.1431	0.8862	
Domestic	17.9469	0.6735	26.6482	<0.0001	0.1561
Dependent Variable: Normalized Total Price					
UK	-0.3686	0.0077	-48.0871	<0.0001	0.3400
DE	-0.0094	0.0119	-0.7868	0.4314	
CA	-0.1752	0.0049	-35.5048	<0.0001	
Time	0.0007	0.0012	0.6119	0.5406	
Domestic	0.1976	0.0075	26.4330	<0.0001	0.1316

Table 6: Regression of Total Prices (Controlling for Book Categories)

In Table 6, total prices, which comprise the unit price added to standard shipping fees, are regressed against the same dummy variables as the prior regression. Again, we find higher prices in the domestic market than in the foreign market. Prices in the United Kingdom and Canada remained lower than the United States market, while there is insufficient evidence of lower prices in Germany at 5% significance level.

Variable	Coefficient	Robust Standard Error	t-Value	Pr > |t|	R^2
Dependent Variable: Time Equalized Price (N = 6247)					
UK	-10.9731	0.6784	-16.1750	<0.0001	0.1628
DE	12.4482	1.1256	11.0590	<0.0001	
CA	-9.8390	0.5330	-18.4581	<0.0001	
Time	-0.0291	0.1128	-0.2578	0.7966	
Domestic	0.4178	0.6309	0.6622	0.5078	0.0455
Dependent Variable: Normalized Time Equalized Price					
UK	-0.1126	0.0082	-13.8181	<0.0001	0.1199
DE	0.1530	0.0126	12.1445	<0.0001	
CA	-0.0996	0.0051	-19.4781	<0.0001	
Time	0.0003	0.0013	0.2593	0.7954	
Domestic	-0.0026	0.0074	-0.3489	0.7272	0.0006

Table 7: Regression of Time Equalized Prices (Controlling for Book Categories)

In Table 7, we utilize the fees associated with expedited shipping as a proxy for waiting time. With a more equal dispatch time, we find that now there is insufficient evidence of higher prices in the domestic market. However, a more country specific analysis shows that prices in the UK and Canadian markets remain lower while prices in the German market are now significantly higher.

Conclusion

There is no available wholesale textbook pricing data to examine if prices are converging to or are already at cost. However, given that wholesale prices do not reflect the total cost incurred by the retailers, the lack of such information is not critical. Instead, we observe the prices borne by the consumer since if the predictions of the Bertrand model hold true, then all retailers earn zero economic profit and prices should accurately reflect the total implicit and explicit costs of the retailers.

Our collected data presents differing results. Simply looking at the unit prices or even the price of a textbook with standard shipping costs included, we find that prices have not converged. Prices are cheaper in Europe than in North America. More detailed analysis shows that textbooks are cheaper in all of the other regions, namely, United Kingdom, Germany and Canada, than in the United States. Interestingly, when we correct for the difference in waiting times owing to different shipment origin locations, we find that prices are not lower in Europe than in North America. While prices are actually higher in Germany than in the United States, retailers in United Kingdom and Canada still charge lower prices.

We understand that this research was carried out at a time when Internet commerce in other geographical regions is still in the infancy stage. Indeed, many other stores have been set up after the price collection was started. Examples include Countrybookstore[6] and Internetbookshop[7]. With an increasing number of stores in total as well as within each region, we can better analyze if the prices are converging globally or

[6] http://www.countrybookstore.co.uk
[7] http://www.internetbookshop.co.uk

only within each geographical region. Future analysis should provide a better indication of whether the Internet is really creating a global, borderless marketplace.

References

"The Big Mac Index", available at: http://economist.com/markets/Bigmac/Index.cfm

A. Marshal, "Principles of Economics", London, 1890, p 325.

Y. Edward, "Economic Consequences of the Internet", 22 Oct 1996, available at: http://www.yardeni.com/public/t_961022.pdf

J. Hitchin, "Bookselling: a prescription for Legislation, Regulation and Freedom", International Workshop on Legislation for the Book World, available at: http://culture.coe.fr/clt/eng/eculiv0.6.html

H. Lee, "Do Electronic Marketplaces Lower the Price of Goods?", Communications of the ACM, Vol. 41 No. 1, January 1998

J. Bailey, "Intermediation and Electronic Markets: Aggregation and Pricing in Internet Commerce", Ph.D. Thesis, MIT, May 1998

J. Bailey and Y. Bakos, "An Exploratory Study of the Emerging Role of Electronic Intermediaries", International Journal of Electronic Commerce, Vol. 1, No. 3, Spring 1997

K. Clay, R. Krishnan and E. Wolff, "Prices and Price Dispersion on the Web: Evidence from the Online Book Industry", March 2001, available at: http://www.heinz.cmu.edu/~kclay/bookpricing.pdf

E. Brynjolfsson and M. Smith, "Frictionless Commerce? A Comparison of Internet and Conventional Retailers", available at: http://ecommerce.mit.edu/papers/friction/friction.pdf

H. Varian, "A Model of Sales." American Economic Review. 70, 4. (1980)

E. Clemons, I. Hann and L. Hitt, "The Nature of Competition among Online Travel Agents: An Empirical Investigation", March 2000, available at: http://grace.wharton.upenn.edu/~lhitt/etravel.htm

D. Miljkovic, "The Law of One Price in International Trade: A Critical Review", Review of Agricultural Economics, Vol. 21, No. 1

Figure 1: Average Normalized Prices by Book Category

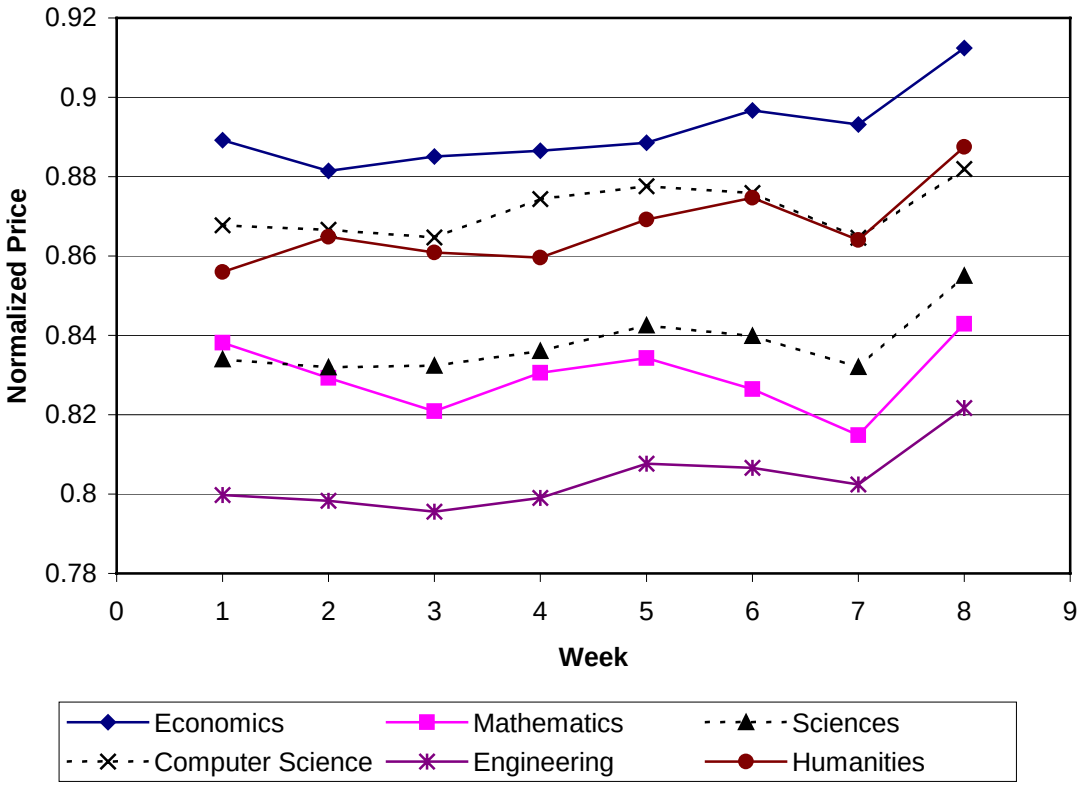


Figure 2: Price Dispersion Over Time

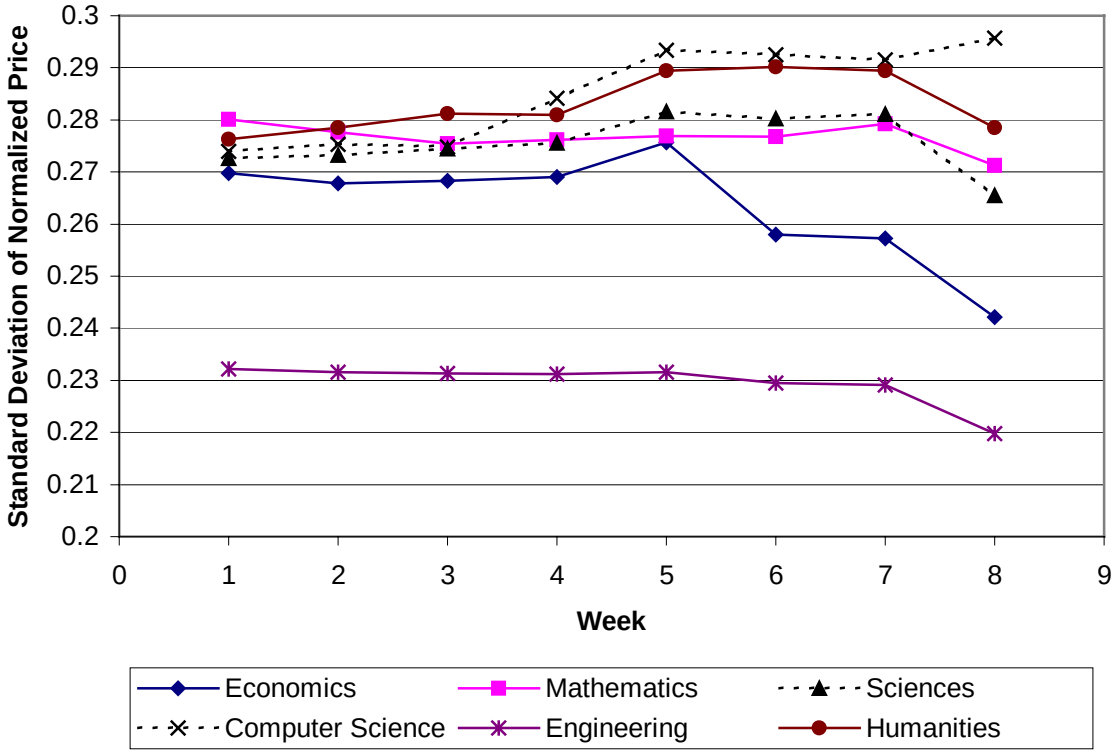
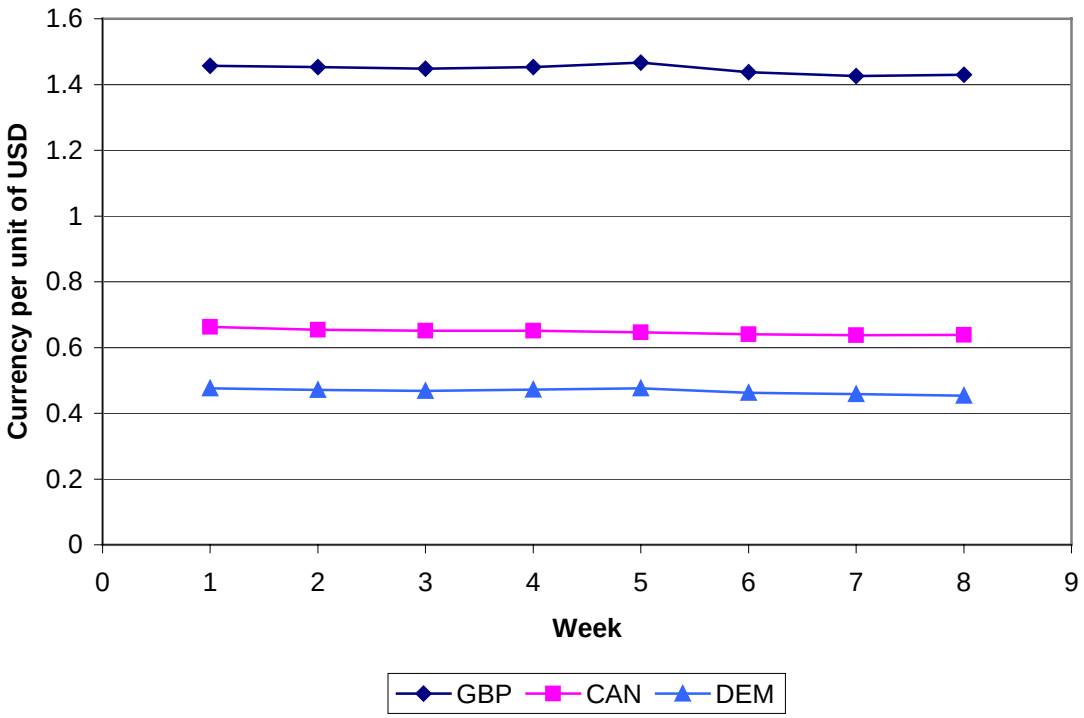


Figure 3: Currency Movements



Java Code

Core files were obtained from Kartik Hosanager with major modifications to the files Seeker.java and Search_Tools.java to reflect different merchant focus and currency representations.

Seeker.java file

```
import java.io.*;
import java.net.*;
import java.sql.*;
import java.util.*;
import java.util.Date;
import java.math.*;
import java.text.*;

public class Seeker extends Search_Book implements Runnable
{


    public Thread thisThread;
    public static int timer_counter = 0;
    public static String book_week = "";
    public static String isbn_file = "";
    public static File output_file;
    public static FileWriter fw;

    void Seeker() {
    }

    public static void main(String[] args) {

      // Check syntax
        if (args.length!=1) {
            System.out.println("Syntax: java seeker [isbn_file_name]");
            System.out.println("Syntax: java seeker isbnlist.txt");
            System.exit(-1);
        };
      // Setup parameter
      isbn_file = args[0];

      // Create a seeker thread and start the thread
      Runnable r = new Seeker();
      Thread myseeker = new Thread(r);
      myseeker.start();
    }



    public static void parse_Result(Vector vect) {
        int i = 0;
        String Keyword = new String("List Price");
        Date now = new Date();
```

```
    String nowDate =

  DateFormat.getDateInstance(DateFormat.SHORT).format(now);
    String nowTime =

  DateFormat.getTimeInstance(DateFormat.SHORT).format(now);

    try {

      for (i=0; i<vect.size(); i++)  {
        String store_data = vect.elementAt(i).toString();
        StringBuffer all_data = new StringBuffer();
        all_data.insert(0,nowTime+"|");
        all_data.insert(0,nowDate+"|");
        all_data.append(store_data);

        // skip "List Price"
        if (store_data.indexOf(Keyword)>0) {
            continue;
        }

        all_data.append("\n");
        fw.write(all_data.toString());

      }  // End of for
      fw.flush();
    } catch (Exception e) {
        System.out.println(e.getMessage());
        System.err.println(e);
    }
}


public  void run() {
    FileReader Isbn_fr;
    BufferedReader Isbn_Input;
    Date file_date;
    String isbn = "";
    String output_filename = "";

    file_date = new Date();
    book_week =

  DateFormat.getDateInstance(DateFormat.SHORT).format(file_date);

  try  {
      // Create output file
        System.out.println("Creating output file...");
        output_filename = book_week.replace('/','-')+"_"+isbn_file;
        output_file = new File(output_filename);
        if (output_file.exists()) {
            System.out.println("Error: Output file already
              exists!");
            System.exit(-1);
        }
        fw = new FileWriter(output_file);
```

```
// Retrieve input file
Isbn_fr       = new FileReader(isbn_file);
Isbn_Input    = new BufferedReader(Isbn_fr);

// Main loop... loop until the end of the input isbn file
int book_count = 0;
int time_out = 0;
while((isbn = Isbn_Input.readLine())!=null)
  {
   // Create threadgroup and store all bookstore thread in it
      ThreadGroup bookGroup =  new ThreadGroup("bookstores");
      Runnable      amz = new amazon();
      Runnable      amd = new amazonde();
      Runnable      bn  = new barnes();
      Runnable      var = new varsity();
      Runnable      chp = new chapters();
      Runnable      idg = new indigo();
      Runnable      auk = new amazonuk();
      Runnable      sbw = new stubookwld();
      Runnable      abs = new alphastreet();

      Thread AMZ = new Thread(bookGroup, amz, "AMZ");
      Thread AMD = new Thread(bookGroup, amd, "AMD");
      Thread BN  = new Thread(bookGroup, bn, "BN");
      Thread VAR = new Thread(bookGroup, var, "VAR");
      Thread CHP = new Thread(bookGroup, chp, "CHP");
      Thread IDG = new Thread(bookGroup, idg, "IDG");
      Thread AUK = new Thread(bookGroup, auk, "AUK");
      Thread SBW = new Thread(bookGroup, sbw, "SBW");
      Thread ABS = new Thread(bookGroup, abs, "ABS");


      book_count++;
      System.out.println("*** current isbn is ---> " + isbn +
                    " *** book count ---> " + book_count);

      Result_Vect.removeAllElements();
      setcurrentISBN(isbn);
      Search_Book.ThreadTotal = 0;

      // Start searching in all online bookstores
      AMZ.start();
      CHP.start();
      IDG.start();
      AUK.start();
      SBW.start();
      AMD.start();
      BN.start();
      VAR.start();
      ABS.start();

      time_out = 0;
      final int unit_time = 10000;// 10 seconds per unit
      final int unit_per_min = 12;
      final int min = 3;
      Thread.currentThread().sleep(unit_time);
      // At most 3 minutes each book
```

```
                while (time_out < unit_per_min*min) {
System.out.println("Current Thread: " +
Thread.currentThread().getName() +
                                   " Time: " + time_out);
                  int numThreads = bookGroup.activeCount();
                  System.out.println("bookGroup.activeCount() =" +
numThreads);

                  if (numThreads<1) {
                      break;
                  }
                  time_out++;
                  Thread.currentThread().sleep(unit_time);
                }
                time_out = 0;
                // Stop threads if time out
                if(bookGroup.activeCount()>1) {

                  if (AMZ.isAlive()) {
                      System.out.println("AMZ is stopped");
                      AMZ.stop();
                  }
                  if (CHP.isAlive()) {
                      System.out.println("CHP is stopped");
                      CHP.stop();
                  }
                  if (IDG.isAlive()) {
                      System.out.println("IDG is stopped");
                      IDG.stop();
                  }
                  if (AUK.isAlive()) {
                      System.out.println("AUK is stopped");
                      AUK.stop();
                  }
                  if (SBW.isAlive()) {
                      System.out.println("SBW is stopped");
                      SBW.stop();
                  }
                  if (ABS.isAlive()) {
                      System.out.println("ABS is stopped");
                      ABS.stop();
                  }
                  if (AMD.isAlive()) {
                      System.out.println("AMD is stopped");
                      AMD.stop();
                  }
                  if (BN.isAlive()) {
                      System.out.println("BN is stopped");
                      BN.stop();
                  }
                  if (VAR.isAlive()) {
                      System.out.println("VAR is stopped");
                      VAR.stop();
                  }

                  System.gc();
                }
```

```
                Search_Book.Thre_Counter = 0;
                Search_Book.ThreadTotal = 0;
                show_Vect(Result_Vect);
                parse_Result(Result_Vect);

              }
            fw.close();
            System.gc();
            System.out.println("Price collecting completed...");
            // Forced exit
            System.exit(0);
      } catch (Exception e)    {
            System.out.println(e.getMessage());
            System.err.println(e);
      }
    }
}

class amazon extends Search_Book
{

    public int session_num = 0;

    amazon()
    {
      super();
    }
    public void setBasic()
    {
      setStore_Name("Amazon");

      BaseURL = "http://www.amazon.com/";
      SearchURL = "http://www.amazon.com/exec/obidos/";
      IsbnSearchURL = SearchURL;
      SessionId = "002-7959661-2416852";
      addShipping("$4.48","3-7 days","Standard");

    }

    public void setSearchString(String isbn)
    {
      SearchString = BaseURL + "exec/obidos/ASIN/" + isbn + "/" +
SessionId;
    }

    public boolean HandleIsbnLine(String line)
    {

      if (grepLine("don't let that stop you",line))
          {
            return false;
          }

      if (grepLine("Our Price:",line))
          {

            addPrice(priceAfter(line, "Our Price:"));
```

```
            return false;
          }

      return true;
    }


}

class barnes extends Search_Book
{

    public int session_num = 0;

    barnes()
    {
      super();
    }
    public void setBasic()
    {
      setStore_Name("BN");

      BaseURL = "http://www.barnesandnoble.com/";
      SearchURL =
"http://shop.barnesandnoble.com/textbooks/booksearch/isbninquiry.asp?";
      IsbnSearchURL = SearchURL;
      addShipping("$4.48","3-7 days","Standard");


  }



  public void setSearchString(String isbn)

  {

      SearchString = IsbnSearchURL + "isbn=" + isbn;

  }



  public boolean HandleIsbnLine(String line)

  {



      if (grepLine("ISBN is not in our database",line))
```

```
            {
                return false;
            }

        if (grepLine("Our New Price:",line))
            {

                addPrice(priceAfter(line, "Our New Price:"));

                return false;
            }

        return true;
    }



}

class varsity extends Search_Book
{

    public int session_num = 0;
```

```
varsity()
{
        super();
}
public void setBasic()
{
        setStore_Name("Varsitybooks");

        BaseURL = "http://www.varsitybooks.com/";
        SearchURL                                                    =
"http://www.varsitybooks.com/detail.asp?dqs=4&aid=370419778&";
        addShipping("$4.95","3-7 days","Standard");

}

public void setSearchString(String isbn)
{
        String newisbn;
        newisbn = isbn.substring(0, 9);
        SearchString = SearchURL + "intProductID=" + code(isbn) + "&framed=true";
}

public boolean HandleIsbnLine(String line)
```

```
    {
        if (grepLine("No detail was found",line))
            {
                return false;
            }

        if (grepLine("Our Price:",line))
            {
                addPrice(priceAfter(line, "Our Price:"));
                return false;
            }
        System.out.println("NOT FOUND AT VAR");
        return true;
    }
}

class stubookwld extends Search_Book
{

    public int session_num = 0;

    stubookwld()
    {
```

```
        super();
}
public void setBasic()
{
        setStore_Name("StudentBookWorld");

        BaseURL = "http://www.studentbookworld.co.uk/";
        SearchURL = "http://www.studentbookworld.co.uk/Results.asp?";
        IsbnSearchURL = SearchURL;
        addShipping("$7.50","7-10 days","Standard GBP");

}

public void setSearchString(String isbn)
{
        SearchString = SearchURL + "sISBN=" + isbn;
}

public boolean HandleIsbnLine(String line)
{
        if (grepLine("don't let that stop you",line))
            {
                return false;
```

```
            }

        if (grepLine("our price",line))
            {
                addPrice(priceAfter(line, "our price £"));
                return false;
            }

        return true;
    }
}

class alphastreet extends Search_Book
{
    public int session_num = 0;

    alphastreet()
    {
        super();
    }
    public void setBasic()
    {
        setStore_Name("AlphabetStreet");
```

```
        BaseURL = "http://www.alphabetstreet.co.uk/";
        SearchURL = "http://www.alphabetstreet.infront.co.uk/cgi-bin/show?Key=alpha";
        IsbnSearchURL = SearchURL;
        addShipping("$3.50","5-21 days","Standard GBP");

}

public void setSearchString(String isbn)
{
        String newisbn;
        newisbn = isbn.substring(0, 9);
        SearchString = SearchURL + newisbn;
}

public boolean HandleIsbnLine(String line)
{

        if (grepLine("NO LONGER AVAILABLE",line))
           {
                return false;
           }
```

```
        if (grepLine("Price: <!--",line))
            {
                addPrice(priceAfter(line, ";"));
                return false;
            }

        return true;
    }
}

class amazonuk extends Search_Book
{

    public int session_num = 0;

    amazonuk()
    {
        super();
    }
    public void setBasic()
    {
        setStore_Name("AmazonUK");
```

```
        BaseURL = "http://www.amazon.co.uk/";
        SearchURL = "http://www.amazon.co.uk/exec/obidos/";
        IsbnSearchURL = SearchURL;
        SessionId = "026-0895523-4536436";
        addShipping("$4.95","8-11 days","Standard GBP");

}

public void setSearchString(String isbn)
{
        SearchString = BaseURL + "exec/obidos/ASIN/" + isbn + "/" + SessionId;
}

public boolean HandleIsbnLine(String line)
{

        if (grepLine("don't let that stop you",line))
            {
                return false;
            }

        if (grepLine("Our Price:",line))
            {
```

```
                out_line = htmlLine(line);
                out_line = takePrice(out_line);
                addPrice(out_line);
                return false;
            }

        return true;
    }
}

class amazonde extends Search_Book
{

    public int session_num = 0;

    amazonde()
    {
        super();
    }
    public void setBasic()
    {
        setStore_Name("AmazonDE");
```

```
        BaseURL = "http://www.amazon.de/";
        SearchURL = "http://www.amazon.de/exec/obidos/";
        IsbnSearchURL = SearchURL;
        SessionId = "028-6180659-1918165";
        addShipping("$34.95","7-21 days","Standard DM");

}

public void setSearchString(String isbn)
{
        SearchString = BaseURL + "exec/obidos/ASIN/" + isbn + "/" + SessionId;
}

public boolean HandleIsbnLine(String line)
{

        if (grepLine("Die von Ihnen",line))
           {
                return false;
           }

        if (grepLine("Preis:",line))
```

```
            {
                addPrice(priceAfter2(line, "DM"));
                return false;
            }

        return true;
    }
}


class chapters extends Search_Book
{
    chapters()
    {
        super();
    }

    public void setBasic()
    {
        setStore_Name("Chapters Canada");

        BaseURL = "http://www.chapters.ca/";
        SearchURL
```

```
        = "http://www.chapters.ca/books/details/default.asp?";

    addShipping("$6","14 days","Standard CAN");
}

public void setSearchString(String Isbn)
{
    SearchString = SearchURL + "ISBN=" + Isbn;
}

public boolean HandleIsbnLine(String line)
{
    if (grepLine("We could not find any titles",line))
        {
            return false;
        }
    if (grepLine("<B>Our Sale Price:",line))
        {
            out_line = takePrice(line);
            addPrice(out_line);
            return false;
        }
    if (grepLine("<B>Our Price:",line))
```

```
            {
                out_line = takePrice(line);
                addPrice(out_line);
                return false;
            }

        return true;
    }
}

class indigo extends Search_Book
{
    indigo()
    {
        super();
    }

    public void setBasic()
    {
        setStore_Name("Indigo.ca Canada");

        BaseURL = "http://www.indigo.ca/";
        SearchURL = BaseURL + "cgi-bin/search.cgi";
```

```
        IsbnSearchURL = SearchURL;

        addShipping("$8.70","14 days","Standard CAN");
}

public void setSearchString(String Isbn)
{
        SearchString =
           "http://www.indigo.ca/cgi-bin/gencgi/search?bn=" + code(Isbn);
}


public int HandleParsingLines2(String line,int method)
{
        String out_line = "";

        switch(method)
           {
           case 0:
                 if (grepLine("Our Price:",line))
                    {
                          return 1;
                    }
```

```
            if (grepLine("0 titles matching your",htmlLine(line)))
                return -1;
            return 0;
        case 1:
            out_line = htmlLine(line);
            out_line = takePrice(out_line);
            addPrice(out_line);
            return -1;
        default:
            return 0;
        }
    }
}
```

Search_Web.java file

```
import java.io.*;
import java.net.*;
import java.sql.*;
import java.util.*;
import java.math.*;

public class Search_Web extends Search_Tools
{
   public boolean     _more       = true;
   public boolean     _found      = false;

   public String SearchURL            = "";
   public String BaseURL              = "";
   public String LongURL              = "";
   public String UserId               = "";
   public String SessionId            = "";
   public String Cookie_Name                  = "";
   public String Cookie_Value         = "";
   public String PostString           = "";
   public String GetString            = "";
   public String SearchString  = "";
```

```
public Vector Parsing_Vect          = new Vector();


public URL                          url;
public PrintStream                  out;
public DataInputStream              in;
public URLConnection                urlconn;
public BufferedInputStream          buffer;
public FileOutputStream             file_out;
public BufferedOutputStream         buffer_out;

public InetAddress                  Iadd;
public Socket                       soc;


public static RandomAccessFile    inFile;
public static RandomAccessFile    outFile;
public static RandomAccessFile    logFile;

public static String  inFileName  = "search_in";
public static String  outFileName = "search_out";
public static String  logFileName = "search_log";
```

```
public StringTokenizer      tok;
public String               line            = "";
public String               out_line        = "";
public String               tokens  = "";
public String               delimits        = "";

private static int      _search_count   = 0;

Search_Web()
{
      if (_search_count == -1)
         {
            try {
               inFile   = new RandomAccessFile(inFileName,"r");
               outFile  = new RandomAccessFile(outFileName,"rw");
               logFile  = new RandomAccessFile(logFileName,"rw");
            }
            catch    (Exception e) {
               System.out.println(e.getMessage());
            }
         }
      _search_count++;
```

```
}

        public  void resetBooleans()
{
        _more        = true;
        _found       = false;
}

public static String code(String name)
{
        return(URLEncoder.encode(name));
}

public static DataInputStream openGet(String someURL)
{
        boolean ddddd = true, pppp = false;
        URL url;
        BufferedInputStream buffer2;

        try {
           url = new URL(someURL);
           buffer2 = new BufferedInputStream(url.openStream());
           return(new DataInputStream(buffer2));
```

```
        }
        catch (MalformedURLException e){
            System.out.println(e.getMessage());
        }
        catch(IOException e){
            System.out.print(e.getMessage()
                + "---------------------------------");
        }
        return(null);
}

public static DataInputStream openPost(String someURL,
        String somePostString)
{

        try {
            URL url     = new URL(someURL);
            URLConnection urlconn = url.openConnection();
            urlconn.setDoOutput(true);
            PrintStream out
                = new PrintStream(urlconn.getOutputStream());
            out.println(somePostString);
            out.close();
```

```
            BufferedInputStream buffer2
                  = new BufferedInputStream(urlconn.getInputStream());
            return(new DataInputStream(buffer2));
        }
        catch(MalformedURLException e){
            System.out.println(e.getMessage());
        }
        catch(IOException e){
            System.out.println(e.getMessage());
        }
        return(null);
}

public static DataInputStream openURL(String someURL)
{
        return(openGet(someURL));
}

        public  DataInputStream openURL()
{
        return(openGet(BaseURL));
}
```

```
public static DataInputStream openSearch(String searchurl,
        String poststring)
{
        if (poststring.equals(""))
            return(openGet(searchurl));
        else
            return(openPost(searchurl,poststring));
}



public static DataInputStream openSearch(String searchurl)
{
        return(openSearch(searchurl,""));
}

public  DataInputStream openSearchStream()
{
        return(openSearchStream(SearchString));
}

public static DataInputStream openSearchStream(String searchurl)
{
        int idx = 0, len = 0;
```

```
        String line1 = "", line2 = "";
        len = searchurl.length();
        if (len <= 0)
                        return(null);
        idx = searchurl.indexOf('|');
        if (idx < 0)
           return(openGet(searchurl));
        if (idx >= len - 1)
           return(openGet(searchurl.substring(0,len-1)));
        if (idx == 0)
           return(openGet(searchurl.substring(1)));
        line1 = searchurl.substring(0,idx);
        line2 = searchurl.substring(idx+1);
        return(openPost(line1,line2));
}

public static DataInputStream openSoc(String addName,String outline)
{
        try {
           InetAddress Iadd = InetAddress.getByName(addName);
           Socket soc = new Socket(Iadd ,80);
           PrintStream out = new PrintStream(soc.getOutputStream());
           BufferedInputStream buffer
```

```
                = new BufferedInputStream(soc.getInputStream());
            out.println(outline);
            soc.close();
            out.close();
            return(new DataInputStream(buffer));
        }
        catch (MalformedURLException e){
            System.out.println(e.getMessage());
        }
        catch (IOException e){
            System.out.println(e.getMessage());
        }
        return(null);
}

public static String getHeader(String addName)
{
        String outline = "";
        String line = "";
        try {
            InetAddress Iadd;
            Iadd = InetAddress.getByName(addName);
            Socket soc = new Socket(Iadd ,80);
```

```
    PrintStream out = new PrintStream(soc.getOutputStream());
    BufferedInputStream buffer
          = new BufferedInputStream(soc.getInputStream());
    DataInputStream in = new DataInputStream(buffer);
    out.println("GET / HTTP/1.0\n");
    while((line = in.readLine()) != null)
          {
             if (line.equals("") || line.equals(" ")
                   || line.equals("\n") || line.equals(" \n"))
                   {
                      break;
                   }
             else
                   outline += "\n" + line;
    }
    in.close();
    soc.close();
    out.close();
    return(outline.substring(1));
}
catch (MalformedURLException e){
    System.out.println(e.getMessage());
}
```

```
        catch (IOException e){
            System.out.println(e.getMessage());
        }
        return("");
}

public  static String getCookie(String addName)
{
        String outline = "";
        String line = "";
        try {
            InetAddress Iadd = InetAddress.getByName(addName);
            Socket soc = new Socket(Iadd ,80);
            PrintStream out = new PrintStream(soc.getOutputStream());
            BufferedInputStream buffer
                  = new BufferedInputStream(soc.getInputStream());
            DataInputStream in = new DataInputStream(buffer);
            System.out.println("***");
            out.println("GET / HTTP/1.0\n");
            while((line = in.readLine()) != null)
                  {
                     if (line.equals("") || line.equals(" ")
                           || line.equals("\n") || line.equals(" \n"))
```

```
                    {
                        break;
                    }
                else
                    {
                        if (line.startsWith("Set-Cookie:"))
                            outline += "\n" + line;
                    }
            }
            in.close();
            soc.close();
            out.close();
            return(outline.substring(1));
        }
        catch (MalformedURLException e){
            System.out.println(e.getMessage());
        }
        catch (IOException e){
            System.out.println(e.getMessage());
        }
        return("");
    }
```

```
public void ParsingWeb(String beg_line, String end_line,
        int method, String searchurl, String poststring)
{
        ParsingWeb(beg_line, end_line, method, searchurl + "|" + poststring);
}

public void ParsingWeb(String beg_line, String end_line,
        int method, String searchurl)
{
        Line_Piece lp = new Line_Piece();
        lp.line = beg_line;
        lp.rest = end_line;
        lp.index = method;
        ParsingWeb(lp,searchurl);
}

public void ParsingWeb(Line_Piece lp, String searchurl, String poststring)
{
        ParsingWeb(lp, searchurl + "|" + poststring);
}

public void ParsingWeb(Line_Piece lp, String searchurl)
{
```

```
        Vector vect = new Vector();
        vect.addElement(lp);
        ParsingWeb(vect,searchurl);
}

public void ParsingWeb(Vector vect, String searchurl, String poststring)
{
        ParsingWeb(vect, searchurl + "|" + poststring);
}

public void ParsingWeb(Vector vect, String searchurl)
{
        int idx = 0, len = vect.size();
        Line_Piece lp   = new Line_Piece();
        Line_Piece relp = new Line_Piece();
        if (len <= 0) return;
        String line = "", rest_line = "";
        String beg_line = "", end_line = "";
        int    method = 0;

        lp = (Line_Piece)vect.elementAt(0);
        beg_line = lp.line;
        end_line = lp.rest;
```

```
method   = lp.index;


DataInputStream in
                = openSearchStream(searchurl);
try    {
   while(idx < len && (line = in.readLine()) != null)
         {
            rest_line += line;
            relp = takePiece(rest_line,beg_line,end_line);
            if (relp.index == 1)
                  {
                     HandleParsingLines(relp.line,method);
                     idx++;
                     if (idx < len)
                           {
                              lp = (Line_Piece)vect.elementAt(idx);
                              beg_line = lp.line;
                              end_line = lp.rest;
                              method   = lp.index;
                           }
                     rest_line = relp.rest;
                  }
            else
```

```
                    rest_line = relp.line;
                }
        }
        catch (Exception e) {
           System.out.println("**************>  "+searchurl);
           System.out.println(e.getMessage());
        }
}

public void ParsingLines(String searchurl)
{
        ParsingLines(searchurl,0);
}

public void ParsingLines(String searchurl, String poststring,int method)
{
        ParsingLines(searchurl + "|" + poststring, method);
}

public void ParsingLines(String searchurl,int method)
{
        boolean _more = true;
        String line = "";
```

```
        DataInputStream in
           = openSearchStream(searchurl);
        try    {
           while(_more && (line = in.readLine()) != null)
                 {
                    _more = HandleParsingLines(line,method);
                 }
        }
        catch (Exception e) {
           System.out.println(e.getMessage());
        }
}

public boolean HandleParsingLines(String line,int method)
{
        switch(method)
           {
           case 0:
                 return(HandleIsbnLine(line));
           default:
                 return(HandleIsbnLine(line));
           }
}
```

```
public boolean HandleIsbnLine(String line)
{
	System.out.println(line);
	return true;
}

public String formatPriceLine(String line)
{
	return(line);
}

public void ParsingLines2(String searchurl)
{
	ParsingLines2(searchurl,0);
}

public void ParsingLines2(String searchurl,String poststring,int method)
{
	ParsingLines2(searchurl + "|" + poststring, method);
}

public void ParsingLines2(String searchurl,int method)
```

```
{
        int idx = method;

        String line = "";
        DataInputStream in
           = openSearchStream(searchurl);
        try    {
           while(idx >= 0 && (line = in.readLine()) != null)
                 {
                    idx = HandleParsingLines2(line, idx);
                 }
        }
        catch (Exception e) {
           System.out.println(e.getMessage());
        }
}

public int HandleParsingLines2(String line,int method)
{
        int idx = method;

        switch(method)
           {
```

```
            case 0:
                HandleIsbnLine2(line);
                return 222222;
            default:
                HandleIsbnLine2(line);
                return 222222;
            }
    }

    public int HandleIsbnLine2(String line)
    {
        return 1;
    }

}
```

Search_Tools.java file

```
import java.io.*;
import java.net.*;
import java.sql.*;
import java.util.*;
import java.math.*;
```

```
public class Search_Tools
{
   private static int      _search_count = 0;
   private int             _tool_id      = 0;

   Search_Tools()
   {
         _tool_id = _search_count;
         _search_count++;
   }

   public int show_tool_id()
   {
         System.out.println(_tool_id);
         return(_tool_id);
   }

   public static String thinLine(String line)
   {
         String outline = "";
         StringTokenizer tok;
         if (line.equals("") || line == null || line.length() <= 0)
```

```
            return line;
        tok = new StringTokenizer(line," \t");
        while(tok.hasMoreTokens())
            {
                outline += tok.nextToken();
                outline += " ";
            }
        return(outline.trim());
}

public static String htmlLine(String line)
{
        return  htmlLine(line,"");
}

public static String htmlLine(String line, String sep)
{
        int len = 0, idx, length = line.length(), lbr, rbr;
        String outline = "", aline = line, token;
        StringTokenizer tok;

        if (line.equals(""))
            return("");
```

```
while(len < length)
   {
        lbr = aline.indexOf('<',len);
        rbr = aline.indexOf('>',len);
        if (lbr >= 0 && rbr < 0)
           {
                if (lbr >= 1)
                   {
                        outline += aline.substring(0,lbr);
                   }
                outline += sep;
                break;
           }
        if (lbr < 0 && rbr >= 0)
           {
                rbr = aline.lastIndexOf('>');
                if (rbr < aline.length() - 1)
                   {
                        aline = aline.substring(rbr+1);
                        length = aline.length();
                        len = 0;
                   }
```

```
            else
                break;
        }
    if (lbr < 0 && rbr < 0)
        {
            outline += aline;
            break;
        }
    if (lbr < rbr && lbr >= 0)
        {
            if (lbr > 0)
                outline += aline.substring(0,lbr);
            outline += sep;
            if (rbr < aline.length() - 1)
                {
                    aline = aline.substring(rbr+1);
                    length = aline.length();
                    len = 0;
                }
            else
                break;
        }
    if (lbr > rbr && rbr >= 0)
```

```
                {
                    token = aline.substring(0,lbr);
                    rbr = token.lastIndexOf('>');
                    aline = aline.substring(rbr+1);
                    length = aline.length();
                    len = 0;
                }
            }
        return(outline);
}


public static String compString(String line, String old,
        String ne, int seper)
{
        int len, len1, len2, len3, i, idx, j, k;
        String outline = "", grep = "yes";

        char fc = old.charAt(0);

        idx = 0; k = 0;
        len1 = line.length(); len2 = old.length(); len3 = ne.length();
```

```
while ((len = line.indexOf(fc,idx)) >= 0 && k == 0)
   {
        for (i = 0; i < len2 && i+len < len1 &&
                 line.charAt(i+len) == old.charAt(i); i++);

        if (i == len2)
           {
                if (seper != 0)
                   {
                        k = 1;
                        return grep;
                   }
                else
                   {
                        if (idx < len) outline += line.substring(idx,len);
                        if (len3 > 0) outline += ne;
                        idx = len + len2;
                   }
           }
        else
           {
                if (seper == 0)
                   {
```

```
                        if (idx <= len) outline += line.substring(idx,len+1);
                    }
                    idx = len + 1;
                }
            }
        if (seper != 0) outline = "no";
        if (seper == 0)
            {
                if (idx < len1) outline += line.substring(idx,len1);
            }
        return outline;
}

public static String replaceLine(String line,String old, String ne)
{
        int len1, len2, len3;

        len1 = line.length(); len2 = old.length(); len3 = ne.length();
        if (len1 == 0 || len2 == 0 ||  len1 < len2)
            return line;
        return (compString(line, old, ne, 0));
}
```

```
public static String replaceleadLine(String line, String old, String nw)
{
      String aline = line, outline = nw;
      if (!line.startsWith(old))
         return aline;
      if (old.length() < aline.length())
         {
               outline += aline.substring(old.length());
         }
      return outline;
}

public static boolean grepLine(String old, String line)
{
      int idx;
      idx = line.indexOf(old);
      if (idx >= 0)
         return true;
      return false;
}

public static char numToChar(int n)
{
```

```
        String word="abcdefghijklmnopqrstuvwxyz";
        if (n >=0 && n <= 25)
           return(word.charAt(n));
        return('0');
}

public static int stringToInt(String line,int beg,int end)
{
        int i, j, k, n = 0;
        int sign = 1;
        k = line.length();
        if (end >= k) end = k-1;
        if (beg < 0) beg = 0;
        for (i = beg; i <= end; i++)
           {
                if (line.charAt(i) >= '0' && line.charAt(i) <= '9')
                   n = 10*n + (line.charAt(i)-'0');
                if (line.charAt(i) == '-')
                   sign = -1;
           }
        return (sign*n);
}
```

```
public static double stringToNum(String line,int beg,int end)
{
    int i, j, k, n = 0,len,sign = 1;
    double db = 0;
    k = line.length();
    if (end >= k) end = k-1;
    if (beg < 0) beg = 0;
    for (i = beg,len = 0; i <= end; i++)
      {
          if (line.charAt(i) >= '0' && line.charAt(i) <= '9')
            {
                n = 10*n + (line.charAt(i)-'0');
                len *= 10;
            }
          if (line.charAt(i) == '.')
            len = 1;
          if (line.charAt(i) == '-' && len == 0)
            sign *= -1;
      }
    db = n;
    if (len > 0) db /= len;
    db *= sign;
    return db;
```

```
}

public static double stringToNum(String line)
{
	return(stringToNum(line,0,line.length()));
}

public static int stringToInt(String line)
{
	return stringToInt(line,0,line.length()-1);
}

public static String intToString(int n)
{
	return String.valueOf(n);
}


public static String takePrice(String line)
{
	char  achar = '0';
	int len, idx, i, length;
	String aline = "";
```

```
if ((idx = line.indexOf('$')) < 0)
   {
        idx = 0;
   }

else
   {
        idx++;
   }

length = line.length();
len = line.indexOf('.');

if (len < 0 || len  > length - 3)
   {
        len = length;
   }
else
   {
        achar = line.charAt(len+1);
        if (achar >= '0' && achar <= '9')
           len += 3;
```

```
            else
               len++;
         }

      if (line.charAt(len-1) == '.')
         len = len - 1;

      for (i = idx; i < len; i++)
         {
            achar = line.charAt(i);
            if ((achar >= '0' && achar <= '9') || achar == '.')
               {
                  aline += achar;
               }
         }
      System.out.println("Current Price found is: " + aline);
      return(aline);
}

public static String takePrice2(String line)
{
      char  achar = '0';
      int len, idx, i, length;
```

```
String aline = "";

if ((idx = line.indexOf('$')) < 0)
   {
        idx = 0;
   }

else
   {
        idx++;
   }

length = line.length();
len = line.indexOf(',');

if (len < 0 || len  > length - 3)
   {
        len = length;
   }
else
   {
        achar = line.charAt(len+1);
        if (achar >= '0' && achar <= '9')
```

```
            len += 3;
        else
            len++;
    }

if (line.charAt(len-1) == ',')
    len = len - 1;

for (i = idx; i < len; i++)
    {
        achar = line.charAt(i);
        if (achar >= '0' && achar <= '9')
            {
                aline += achar;
            }
        if (achar == ',')
            {
                aline += '.';
            }

    }
System.out.println("Current Price found is: " + aline);
return(aline);
```

```
}

public static float priceToNum(String line)
{
        float pri = 0;
        char  car = ' ';
        int   i, len, n, m;

        len = line.length();
        for (i = 0,m = 0, n = 0; i < len; i++)
           {
                car = line.charAt(i);
                if (car == '.')
                   m = 1;
                if (car >= '0' && car <= '9')
                   {
                        n = 10*n + (car - '0');
                        m = 10*m;
                   }
           }
        pri = n;
        if (m > 0)
           pri /= m;
```

```
        return pri;
}

public static void addSearch(Vector vect,String line, String rest, int idx)
{
        Line_Piece lp = new Line_Piece();
        lp.index = idx;
        lp.line = line;
        lp.rest = rest;
        vect.addElement(lp);
}

public static Line_Piece takePiece(String line, String beg, String end)
{
        int idx, len, k;
        Line_Piece ls = new Line_Piece();
        if (beg.equals(""))
           {
                if (end.equals(""))
                   {
                        ls.index = 1;
                        ls.line = line;
                        ls.rest = "";
```

```
                return ls;
            }
        len = line.indexOf(end);
        if (len < 0)
            {
                ls.index = -1;
                return ls;
            }
        ls.line = line.substring(0,len);
        ls.line += end;
        ls.index = 1;
        k = end.length();
        ls.rest = line.substring(len+k);
        return ls;
    }
if (end.equals(""))
    {
        idx = line.indexOf(beg);
        if (idx >= 0)
            {
                ls.line = line.substring(idx);
                ls.index = 1;
            }
```

```
            return ls;
        }
    idx = line.indexOf(beg);
    if (idx < 0)
        {
            ls.index = -1;
            return ls;
        }
    len = line.indexOf(end);
    if (len < 0)
        {
            idx = line.lastIndexOf(beg);
            if (idx >= 0)
                {
                    ls.line = line.substring(idx);
                }
            ls.index = -1;
            return ls;
        }
    k = line.lastIndexOf(end);
    if (k < idx)
        {
            ls.index = -1;
```

```
                return ls;
            }
        ls.line = line.substring(idx);
        len     = ls.line.indexOf(end);
        k = end.length();
        ls.rest = ls.line.substring(len+k);
        ls.line = ls.line.substring(0,len);
        k = ls.line.lastIndexOf(beg);
        ls.line = ls.line.substring(k);
        ls.line += end;
        ls.index = 1;
        return ls;
}


public static void show_Vect(Vector vect)
{
        int len = vect.size(), i = 0;
        for (i = 0; i < len; i++)
            {
                System.out.println((String)vect.elementAt(i));
            }
}
```

```
public static String priceAfter(String line,String OurPrice)
{
	String out_line = "";
	int idx = 0;
	out_line = line;
	idx = out_line.indexOf(OurPrice);
	if (idx >= 0)
	   {
		out_line = out_line.substring(idx+1);
		out_line = takePrice(out_line);
		if (!out_line.equals("0.00") && !out_line.equals(""))
		   return out_line;
	   }
	return ("");
}



public static String priceAfter2(String line,String OurPrice)
{
	String out_line = "";
	int idx = 0;
	out_line = line;
```

```
        idx = out_line.indexOf(OurPrice);
        if (idx >= 0)
           {
                out_line = out_line.substring(idx+1);
                out_line = takePrice2(out_line);
                if (!out_line.equals("0.00") && !out_line.equals(""))
                   return out_line;
           }
        return ("");
   }

}
```

Search_Book.java file

```
import java.io.*;
import java.net.*;
import java.sql.*;
import java.util.*;
import java.math.*;

public class Search_Book extends Search_Web implements Runnable
{
```

```
public static int       Thre_Counter = 0;

public static Thread [] ThreadArray= new Thread[50];
public static int       ThreadTotal= 0;


public static Vector    Result_Vect = new Vector();
public static Hashtable Result_Hash = new Hashtable();
public static Hashtable Store_Hash  = new Hashtable();

public static String StandardString = "";
public static String currentISBN    = "";

public String ISBN                  = "";
public String IsbnSearchURL                 = "";
public String PriceString           = "";
public String SearchString                  = "";

public Vector Price_Vect            = new Vector();
public Vector Shipping_Vect         = new Vector();

public String Store_Name    = "";
```

```
public int Store_Id         = 0;

public String Store_Type    = "";
public String Store_Logo    = "";
public int Interface_Type   = 0;

public static int _book_count         = 0;

Search_Book()
{
        super();
        Store_Id = _book_count;
        setBasic();
        _book_count++;
}

public void setBasic()
{
}


private static synchronized void incCounter() {
    Thre_Counter++;
```

```
        ThreadTotal++;
}

private static synchronized void  decCounter() {
   Thre_Counter--;
}

private static synchronized void resetCounter() {
   Thre_Counter=0;
}

public void run()
{
       setSearchString(currentISBN);
       ISBN = currentISBN;
   incCounter();
       System.out.println("starting search -> " + Store_Name);
       execSearch();
   decCounter();
       System.out.println();
       System.out.println("--------End search -> " + Store_Name +
                  " Thread number :" +
                  Thread.currentThread().getThreadGroup().activeCount());
```

```
}

public void execSearch()
{
    execSearch(SearchString);
}

public void execSearch(String searchurl)
{
    if (HandleParsingLines2("test string",10000) != 222222)
      {
          ParsingLines2(searchurl,0);
      }
    else
      {
          ParsingLines(searchurl,0);
      }
}

public int HandleParsingLines2(String line, int method)
{
    return(222222);
```

```
}

public boolean takeIsbn(String isbn)
{
    if (isbn.trim().length() != 10)
      {
          return false;
      }
    ISBN = isbn.trim();
    setSearchString(ISBN);
    return true;
}

public static void setcurrentISBN(String isbn)
{
    if (isbn.trim().length() != 10)
      {
          return;
      }
    currentISBN = isbn.trim();
}

public void setSearchString(String isbn)
```

```
{
}

public void setSearchString()
{
    setSearchString(currentISBN);
}

public static void show_Result()
{
    show_Vect(Result_Vect);
}

public  synchronized void addResult()
{
    addResult(Price_Vect);
}


public static synchronized void addResult(String line)
{
    Result_Vect.addElement(line);
}
```

```
public static synchronized void addResult(Vector vect)
{
	int len = vect.size(), i = 0;
	for (i = 0; i < len; i++)
	  {
		Result_Vect.addElement((String)vect.elementAt(i));
	  }
}

public String formatPriceLine(String line)
{
	return(line);
}

public void notFound()
{
	System.out.println("N/A");
}

public void setStore_Name(String name)
{
	Store_Name = name;
```

```
        Store_Hash.put(name,String.valueOf(Store_Id));
}

public void addShipping(String COSTs, String DAYs, String Serv)
{
        addShipping(COSTs, DAYs, Serv,"N/A");
}

public void addShipping(String COSTs,String DAYs, String Serv,String NA)
{
        Shipping_Vect.addElement(COSTs + "|" + DAYs + "|" + Serv + "|" + NA);
}

public synchronized void addPrice(String line)
{
        int idx = 0, len = Shipping_Vect.size();
        float dol = 0, cst = 0;

        StringTokenizer tok;
        String token = "", totPrice = "",out_line = "",shp = "";

        dol = priceToNum(line);
        for (idx = 0; idx < len; idx++)
```

```
            {
                out_line = ISBN + "|" + StandardString + "|";
                token = (String) Shipping_Vect.elementAt(idx);
                tok = new StringTokenizer(token,"|");
                token = tok.nextToken();
                shp   = "$" + String.valueOf(priceToNum(token));

                totPrice
                   = "$"+ takePrice(String.valueOf(priceToNum(token)+dol));

                out_line += totPrice+"|" + Store_Name +
                   "|$" + String.valueOf(dol) + "|$0.00|" + shp;
                while(tok.hasMoreTokens())
                   {
                        out_line += "|" + tok.nextToken();
                   }
                Result_Vect.addElement(out_line);
            }
        }
    }
```

Line_Piece.java file

```
import java.io.*;
import java.net.*;
import java.sql.*;
import java.util.*;
import java.math.*;

public class Line_Piece
{
   public String line = "";
   public String rest = "";
   public int    index = -1;

   public Line_Piece()
   {
         this.line = "";
         this.rest = "";
         this.index = -1;
   }
}
```