Format Conversion And Report Viewer

A project done at nuva Systems Pvt. Ltd., Coimbatore

PROJECT REPORT

Submitted in partial fulfilment of the requirements
for the award of the Degree of
MASTER OF ENGINEERING
of **Bharathiar University**

Submitted by

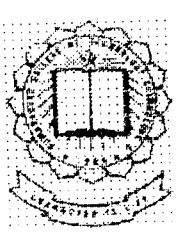
K.UMAMAHESWARI
Reg.No.: 9937K0014

Internal Guide

Mr. R.Dinesh B.Tech., M.S.(USA),
Senior Lecturer,Dept. of Comp.Science
Kumaraguru College of Technology, Coimbatore

External Guide

Miss. V.S.Prabhaa, M.C.A
Software Engineer, nuva Systems Pvt.Ltd.,
Coimbatore



Department of Computer Science and Engineering
Kumaraguru College of Technology
Coimbatore-641 006.
January 2001

CERTIFICATE

This is to certify that the project work entitled

"FORMAT CONVERSION AND REPORT VIEWER"

submitted to the

Department of Computer Science and Engineering

Kumaraguru College of Technology

in partial fulfilment of the requirements for the award of the Degree of Master of Engineering is a record of original work done by Mrs. Umamaheswari. K. Reg.No: 9937K0014 during her period of study in the Department of Computer Science and Engineering**, Kumaraguru College of Technology** ,Coimbatore under my supervision and this project work has not formed the basis of award of any Degree/Diploma / Associateship/Fellowship or similar title to any candidate of any university.

S. Thangasamy 12/1/

Professor and Head

R. Dinesh (11/2001)

Staff-in-charge

Submitted to University Examination held on 23/1/2001

R. Dinesh (2011/2001)

Internal Examiner

External Examiner 23/1/2001

SYSTEMS

8th January, 2001

TO WHOMSOEVER IT MAY CONCERN

This is to certify that ***Mrs.K.UMA MAHESWARI*** ,doing Final Year ME (Computer Science & Engineering) at Kumaraguru College Of Technology has done her project in our organization.

Title : **FORMAT CONVERSION AND REPORT VIEWER**

Technology used : **Java**

Duration : **Six Months (July – December 2000)**

During the period her conduct was good.

For Nuva systems (P) Ltd

D.RaviChandran
(Projects Manager)

nuva systems ltd.
"THE UFFIZI", 338, Avanashi Road • Coimbatore - 641 004 • India

DECLARATION

I hereby declare that this project work entitled

"Format Conversion and Report Viewer"

submitted to Kumaraguru College of Technology, Coimbatore (Affiliated to Bharathiar University) is a record of original work done by me under the supervision and guidance of Mr.R.Dinesh, M.S., Senior Lecturer, Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore under my supervision and guidance and his project work has not formed the basis of award of any Degree / Diploma / Associateship / Fellowship or similar title to any candidate of any University.

K. Umamaheswari

Place: Coimbatore.
Date : 11/1/2001

Signature of Candidate
(K. Umamaheswari)

R. Dinesh (11/1/2001)

Counter signed by

Staff-in-charge

Mr.R.Dinesh,
Senior Lecturer,
Department of Computer Science and Engg.
Kumaraguru College of Technology,
Coimbatore.

DEDICATED TO MY FAMILY AND AUNTY

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

I take this opportunity to express my sincere gratitude to the management for providing all the required facilities in our college for completing the project.

I gratefully extend my hearty thanks to our revered Principal **Dr. K.K. Padmanabhan,** B.Sc.(Engg.,)M.Tech., Ph.D.,FIE,FI Mech E FII Prod E, FIV, MISTE,MIIIE.,and to our beloved Head of the department, **Prof. S. Thangasamy,** Ph.D for their valuable suggestions and constructive criticisms.

My grateful thanks to my internal guide **Mr. R. Dinesh,** M.S.,Senior Lecturer for his constant support and guidance throughout my project.

My sincere thanks to our class advisor **Assistant Professor Mr. R. Kannan,** M.E., and to all the staff members of the computer department, who have rendered their help with the greatest enthusiasm and interest.

I express my sincere thanks to Mr.D. Ravichandran, Projects Manager at nuva Systems Pvt. Ltd., for providing full support and guidance in completing the project.

I take pleasure in thanking my external guide Ms. V.S.Prabha M.C.A., nuva Systems who gave me a strong support and inspiration in the development of the project.

I also thank Mr.N.NandaKumar B.E., Java Faculty at nuva Systems who had helped me in the development of the project and I thanks all of nuva systems. I extend my thanks to Ms.Supriya, B.E.,for her timely help.

I express my sincere gratitude to the Managing Director,Mr. Sridhar Varadharaj, nuva Systems Pvt. Ltd., for providing me the opportunity to do this project in their esteemed organization.

Finally, I feel happy in thanking my family, Granny, Aunty Ms C. Radha and my parents who has given me a valuable support.

I thank my friends and all who helped me directly or indirectly to complete this project work successfully.

Umamaheswari.K

CONTENTS

INTRODUCTION

SYNOPSIS

SYNOPSIS

In this communication world, the information is spread through different communication methods and media's using the technologies like media broadcasting and satellite communication. So information are send in different formats which need to be format converted for report viewing according to the users need.

Generally in applications like Visual Basic, etc, the out put format is in Text (.txt) format, Rich Text Format (.rtf) format and so on. For these, we have editors like MS Word to edit and manipulate the information in those files.

Whereas when we come to Web applications, the Web Browser displays only the HTML file format and the Users do not have the flexibility to view, search, format and print the text contents in the browser.

Now a days output data are stored as rtf format, in order the utilize the usage of keeping the data in rtf format. So it is an important thing to consider, how to handle those data for MIS, EIS, etc... through web.

So, a software tool is developed that will load these type of output files and give all the above said flexibilities to the user or the viewer.

We go for java as the development software, since it is platform independent and most widely used for Internet based applications.

The design aspect of this software provides user – friendliness and richness while **viewing report** through the web browser type of files supported: rtf, txt, HTML, etc.

INTRODUCTION

Internet has dominated the world of computer communication where information are sent in different formats as RichText Format, HyperText Markup Language, etc .Users need them format converted to view the information in the normal text format. So this software is being developed to convert the RTF and HTML format to normal text and is viewed by the report viewer that is also developed.

ABOUT THE SYSTEM

OBJECTIVE

Generally in applications like Visual Basic, etc, the out put format is in Text (.txt) format, Rich Text Format (.rtf) format and so on. For these, we have editors like MS Word to edit and manipulate the information in those files.

Whereas when we come to Web applications, the Web Browser displays only the HTML file format and the Users do not have the flexibility to view, search, format and print the text contents in the browser.

Now a days output data are stored as RTF format, in order the utilize the usage of keeping the data in RTF format. So it is an important thing to consider, how to handle those data for MIS, EIS etc... through web.

So, we need a software tool that will load these type of output files and give all the above said flexibilities to the user or the viewer. We go for Java as the development software, since it is platform independent and most widely used for Internet based applications.
The design aspect of this software provides user – friendliness and richness while **viewing report** through the web browser type of files supported: rtf, txt, HTML, etc.

SOFTWARE SPECIFICATIONS

DESCRIPTION:

To develop a Software that can be utilized to convert the Web Browser types of files to the required format. All files in the Web browser are of html nature, which would be developed in any format like text, doc, rtf, etc., which the User for MIS needs them to be format converted. So an interface is developed between the Web browser and the Beans environment, such that all the required files in the Web browser are converted to the Beans environment, where the downloaded files could be used like an editor.

The file is first received and recognized for identification. The file when converted to any format can be viewed by any user according to his need. Various optional capabilities provide the User in generating a report or altering the text in the required font, color, etc.

These event-driven options are capable of quick access and alterations are performed in no time.

Since the software is developed in Java application, it can be utilized for any files residing in different Operating Systems like DOS, UNIX, WINDOWS based files, etc.

A Java Bean is a software component that will be designed to be reusable in a variety of different environments. It may be used to perform a single function or a complex one depending on the requirement in future.

A Bean may be designed to work autonomously on a Users workstation or to work in cooperation with a set of other distributed components.

When working with Java Beans, the Developers use an Application Builder Tool, a utility that enables one to configure a set of Beans, connect them together and produce as working application.

The Bean Developer Kit (BDK) is one such utility that enables one to create configure and connect a set of Beans.

ADVANTAGES :

Generally, the files are not user–friendly. It could be viewed as it is sent or presented in the web. But loading the software developed into the system, all the utilities are provided in such a way that it satisfies the users requirement without affecting the Base data or information residing in the file.

It also reduces the search time on request of a specified area or portion in spontaneous access. It is easy to search any page or part of the application downloaded which is not possible in normal browsing.

This software could be enhanced to convert any files apart from rtf, text, html also.

JAVA-THE LANGUAGE

LANGUAGE JAVA AND OPERATING SYSTEMS

Learning a leading edge language (Java) and a leading edge programming paradigm (Object oriented programming) head into a world in which the internet has a massive new prominence.

Java was promoted in 1995 as a means of adding dynamic content to World Wide Web pages. Instead of web pages with only text and static graphics, people's web pages 'come alive' with audios , animation's interactivity and so on, video and three dimensional imaging.

These features are precisely what businesses and organisations really need to meet today's information processing requirements. So, Java is viewed as having the potential to become one of the world's key general-purpose programming languages.

Keep the language concise by removing special-purpose features that are used by only small segments of the programming community.

The computer field has never seen anything like the Internet / world wide web / Java "explosion" occurring today. People want to communicate. People need to communicate. Sure they have been doing that since the dawn of civilization, but computer communication has become important passing information back and forth.

Language is truly portable so it is appropriate for implementing Internet-based application and web-based application. Java works well on platforms including those for sun's solaris, dos, unix and microsoft's window '95 and windows NT.

Java is sure to play a key role in making many of the possibilities become reality. There is great stuff to be done in Java so let's get it right mean. Java is not trivial by any means but it's fun and greatest advantage it is a reusable component.

ABOUT FORMAT CONVERSION

ABOUT FORMAT CONVERSIONS

INTRODUCTION

Format Conversion applications take a file in one format, and convert it to another. From the very beginning of computing there were no standards sets and most computers were basically huge machines to perform calculations upon.

After some time someone realized that computers could be used to keep and format texts and moreover print it on paper. Unfortunately, the computer manufacturers had their own minds about how to code the different characters on their machines. Thus came a long series of different programs to convert from one character set to another. The American Standards Association set up a goal of generating a universal character set and in 1963 the American Standard Code for Information Interchange (*ASCII*) was born .

To facilitate type setting and editing of documents word processors came along. There were at least one word processor created for every major brand of computers, with some having much more than one. The main problem with the word processors were that they did not know how to communicate with each other, since every word processor had its own way of storing information on disc.

There are a few type setting programs available on more than one platform, but these are unfortunately not as easy to use for a beginner as the what-you-see-is-what-you-get(WYSIWYG) approach of the word processors. The main contenders in this group of type setters were and are the roff family (nroff, troff, groff, tbl, eqn, mm, etc.), the tex family (TeX, LaTeX, etc.) and SGML

Microsoft Corporation defined yet another format in 1989 called Rich Text Format (*RTF*). This has been further expanded in 1994. The main differences between this format and the earlier word processor formats is that it is open for anyone to use and a pure text format so that it is easier to take files from one machine to another.

One reason for using one of the pure text representations is that it can quite easily be transformed into another format by the use of a filter. Such filters have been created many times before, one of them being in a troff to TEXTNET translator.

The technology in the WWW is evolving incredibly rapidly; it is certain that lots more converters will be available soon in the market with much more powerful functionality. WAIS and WAISGATE are used for creating the database. If there are many documents, and you are aware of the nature of the documents (e.g all ACS publication), then it is best to search documents and view them using the WWW clients (Mosaic, Netscape,...).

Commonly used document formats are as follow:

- Plain ASCII text (.TXT)
- Document format specific to the authoring software: MS Word (.DOC), MS Excel (.XLS), MS PowerPoint (.PPT), FrameMaker format, PageMaker format, TeX/LaTeX (.TEX) and .DVI format, etc.
- Graphics formats: GIF, JPEG, etc.
- Printing language formats - Postscript/Encapsulated Postscript, PCL, HPGL, etc.
  AdobePortableDocumentFormat(.PDF).

Some formats are better than others for a particular purpose, graphics rich document needed to be treated differently from a plain text one.

Format conversion facilities must be available to allow simple browsers to access data which is stored in a sophisticated format. The basic format is a set of formats which every client must be able to handle. These include 80-column text.

ENSURES UNIVERSAL READABILITY

A server providing a format which is not in the basic set of formats required for a client must have the possibility of generating some sort of conversion of the text for a client which cannot handle it.

When a server (or browser) is obliged to perform a conversion from one format to another one imagines that the result would be cached so that if the same conversion were needed later, it would be available more rapidly.

Format conversion is something which can be triggered either by the writer or by the browser. Conversion takes place immediately when they are entered and kept until the source has been updated.

NECESSITY:

There is increasing concern in the research community of the need to standardise on formats for electronic document interchange. There are a growing number of word processing formats that complicate an already complex array of proprietary and de facto multi-media standards. The project proposes to identify and investigate the leading word processing formats and conversion tools with an aim of providing a solution to multi-format and multi-platform document interchangeability.

The aim is not only to study possible packages, but also to demonstrate such interchangeability both in electronic mail and database access environments.

Format Conversion Applications

With the growth of Internet technologies, distribution of documents/material in electronic forms via web servers offer both

convenience and quality. Within university environment, putting up lecture notes, assignments, lab materials and research papers on web servers for downloading are getting popular, and it helps to improve the way people teach, learn and collaborate.

Electronic document delivery from a database involves an author generating documents in a proprietary format and providing the electronic form to organisations running a database service. These will then provide the documents to the users by storing them in a database in the original format or in popular used format by the user.

The user for these documents will require them in one of two ways. Either they will be requested in their original format; then they will need the tools of the originator of the documents for reading and browsing. Alternatively, they will wish to search against a collection of documents, using some kind of query mechanism, and then browse or read the articles that were found. It should be possible also that a reader browse the document in his/her favourite format using the tools available in-house.

An on-line database of Electronic Documents offers many advantages over the conventional paper-based Documents; many of these advantages fall into the areas of search and access. Searching texts electronically is much easier than manually.

The Word Wide Web (WWW) [3] is the largest information service on the Internet - and probably in the world. WWW is a client-server

system with both clients and servers throughout the world. A WWW server is a program running on a computer that listens on a TCP port for incoming connections from WWW clients. It expects a connecting client to speak in a protocol called Hypertext Transfer Protocol (HTTP) [3].

The documents in the WWW are written in HTML (Hypertext Mark-up Language) format, and delivered from WWW servers to the Clients in this form. The technology in the WWW is evolving incredibly rapidly; many more proprietary format to HTML converters are needed. Hence the importance of WWW access to documents.

Features of HTML and RTF

The script allows user to prepare documentation in any text-oriented formats (now supported RTF and HTML) by providing text resources those describe conversion between the internal style names and the output formatting directives.

For instance, if you wish to use "Heading 1" formatting style in the documentation, you should add the new resource for the output language:

HTML example:	*RTF example:**
Heading_1_start = <H1>	Heading_1_start = \\s1 {

Heading_1_end = </H1>	Heading_1_end = \\par}

So when the style is used in the script for a manner like the following:

outputStream.print (myStyles.style ("Heading 1") + "*My heading text*"
+ myStyles.styleEnd ("Heading 1"));

The script will generate the text according to the output format:

For HTML:	*For RTF :**
<H1>My heading text</H1>	\\s1 {My heading text \\par}

The general aims of the project are as follows :

- Investigate the transfer of documents.
- Investigate the storage and retrieval of compound documents in a searchable form.
- Set up of a User-friendly interface for editing documents from formats like HTML/RTF form to our format and storing it in that format.
- Processable form, allowing the processing of the document.
- Formatted form, allowing the presentation of the document as intended by the document author.

- Formatted Processable form, allowing both presentation and processing.
- Any paragraph within the body that's not explicitly tagged is Normal Text.
- Word-wrapped by the client, depending on the set column size and font sizes

HTML TO PLAIN TEXT CONVERTER

HTML TO PLAIN TEXT CONVERTER

FORMAT USED IN WEB:

It's easier and more reliable to create a sort of programming language that allows them to *describe* the Web page. The particular language used by the WWW is called **HTML**, which stands for *hypertext markup language*

HTML is a member of the SGML family of markup languages

Web authors work with HTML. Even if your source files came from another source, once they have been marked up using HMTL, they are no longer viewable without an HTML browser. So, it is often necessary to convert an HTML document back to plain text.

CONVERSION:

Simple approach would be to simply remove all the HTML markup from a file. This would leave a unformatted text file containing many extra spaces, tabs, and line breaks.

A better approach would be to interpret the HTML tags contained in the document, much as an HTML browser does, and create a text file containing some of the formatting from the original.

Options should be provided to allow some control of the how the text file is rendered, including choosing word-wrap, setting maximum line length, and choosing to render horizontal rules (<HR> tags) or not.

Most HTML tags look like:

```
<TheTagName> affected text </TheTagName>
```

The tag name itself (here, `TheTagName`) is enclosed in brackets (<>). HTML tags generally have a beginning and an ending tag, surrounding the text that they affect.

The beginning tag "turns on" a feature (such as headings, bold, and so on), and the ending tag turns it off. Closing tags have the tag name preceded by a slash (/).

New Term
HTML tags are the things inside brackets (<>) that indicate features or elements of a page.

STANDARD STRUCTURE OF HTML:

```
<html>
  <head>
    <title>The document title
    </title>
  </head>
     <body>
          Text and markup
      <address>
            Author and version info
      </address>
```

```
        </body>
</html>
```

with all the general basic tags which need to be upload for web applications. When the user wants to view the information in his format need to convert by removing the tags.

STRUCTURE OF TEXT AFTER CONVERSION

- The document title

  Text and markup

  Author and version info

-

Which the user can alter the format using the report viewer developed.

The purpose of the HTML parser is to retrieve specific information from an HTML page. Before you can extract data from an HTML page, first to understand HTML.

HTML is composed of text and formatting that appear as HTML "tags".
For example <B> and </B> make something bold, as follows:
<B> Bold Title </B>

But we are concerned with only the actual text and not the formatting information and building a set of short character strings. We strip out all the HTML tags and retrieve the value from the very next token.

To do this, we are going to have a method to give the HTMLParser an HTML page to parse as well as methods to retrieve tokens and token information. Separate class methods for HTMLtokenizer is used to parse the html web based files and are being parsed to take the text alone. It is being stored in a separated file after reading from the net. Then parsed and made to display from the file to our editor.

Tables 1 and 2 show the properties and methods for the HTMLParser class respectively,

Table 1 HTMLParser Properties

Property Name	Description
HTML page	Stores the text of the HTML page for later parsing -if we want to add any other kind of parsing (extract links or graphic filenames?)
Token Collection	A collection of post-parsed HTML page tokens
Number of Tokens	The number of tokens in the collection

Table 2 HTMLParser Methods

Method Name	Description
ParsePage(String)	Parses the HTML page contained in the String into tokens
FindToken(String)	Returns the index of the token that matches the string passed to the methods

There are the following problems with the HTML conversions:

- Automatic section numbering is lost in the HyperText document in HTML
- Multi-section properties are ignored.
- Centre text are ignored in the HTML
- Indentation are ignored in the HTML
- Graphics are ignored (HTML does not support Graphics)

RTF TO PLAIN TEXT CONVERTER

RTF TO PLAIN TEXT CONVERTER

INTRODUCTION

The Rich Text Format (RTF) Specification is a method of encoding formatted text and graphics for easy transfer between applications. Currently, users depend on special translation software to move word-processing documents between different MS-DOS®, Windows, OS/2, Macintosh, and Power Macintosh applications.

The RTF Specification provides a format for text and graphics interchange that can be used with different output devices, operating environments, and operating systems. RTF uses the ANSI, PC-8, Macintosh, or IBM PC character set to control the representation and formatting of a document, both on the screen and in print. With the RTF Specification, documents created under different operating systems and with different software applications can be transferred between those operating systems and applications. RTF files created in Word 6.0 (and later) for the Macintosh and Power Macintosh have a file type of "RTF."

Software that takes a formatted file and turns it into an RTF file is called a writer. An RTF writer separates the application's control information from the actual text and writes a new file containing the

text and the RTF groups associated with that text. Software that translates an RTF file into a formatted file is called a reader.

Abstract

The Rich Text Format (*RTF*) by Microsoft Corporation has lately become a *de facto* standard format, portable between computer systems. This work gives a brief overview of the grammar of the format, as well as outlining the first few step towards a format converter from *RTF* to text.

RTF Syntax

An RTF file consists of unformatted text, control words, control symbols, and groups. For ease of transport, a standard RTF file can consist of only 7-bit ASCII characters. (Converters that communicate with Microsoft Word for Windows or Microsoft Word for the Macintosh should expect 8-bit characters.) There is no set maximum line length for an RTF file.

A *control word* is a specially formatted command that RTF uses to mark printer control codes and information that applications use to manage documents. A control word cannot be longer than 32 characters. A control word takes the following form:

\LetterSequence<Delimiter>

Note that a backslash begins each control word.

The LetterSequence is made up of lowercase alphabetic characters between "a" and "z" inclusive. RTF is case sensitive, and all RTF control words must be lowercase.

The delimiter marks the end of an RTF control word, and can be one of the following:

- A space. In this case, the space is part of the control word.
- A digit or a hyphen (-), which indicates that a numeric parameter follows. The subsequent digital sequence is then delimited by a space or any character other than a letter or a digit. The parameter can be a positive or a negative number. The range of the values for the number is generally –32767 through 32767. However, Word tends to restrict the range to –31680 through 31680. Word allows values in the range -2,147,483,648 to 2,147,483,648 for a small number of keywords (specifically **\bin**, **\revdttm,** and some picture properties).

- An RTF parser must handle an arbitrary string of digits as a legal value for a keyword. If a numeric parameter immediately follows the control word, this parameter becomes part of the control word. The control word is then delimited by a space or a nonalphabetic or nonnumeric character in the same manner as any other control word.

- Any character other than a letter or a digit. In this case, the delimiting character terminates the control word but is not actually part of the control word.

If a space delimits the control word, the space does not appear in the document. Any characters following the delimiter, including spaces, will appear in the document. For this reason, you should use spaces only where necessary; do not use spaces merely to break up RTF code.

A *control symbol* consists of a backslash followed by a single, nonalphabetic character. For example, \~ represents a nonbreaking space. Control symbols take no delimiters.

A *group* consists of text and control words or control symbols enclosed in braces ({ }). The opening brace ({) indicates the start of the group and the closing brace (}) indicates the end of the group. Each group specifies the text affected by the group and the different attributes of that text.

The RTF file can also include groups for fonts, styles, screen color, pictures, footnotes, comments (annotations), headers and footers, summary information, fields, and bookmarks, as well as document-, section-, paragraph-, and character-formatting properties. If the font, file, style, screen-color, revision mark, and summary-information groups and document-formatting properties are included, they must

precede the first plain-text character in the document. These groups form the RTF file header.

If the group for fonts is included, it should precede the group for styles. If any group is not used, it can be omitted. The groups are discussed in the following sections.

The control properties of certain control words (such as bold, italic, keep together, and so on) have only two states.

When such a control word has no parameter or has a nonzero parameter, it is assumed that the control word turns on the property.

When such a control word has a parameter of 0 , it is assumed that the control word turns off the property. For example, **\b** turns on bold, whereas **\b0** turns off bold.

Certain control words, referred to as *destinations*, mark the beginning of a collection of related text that could appear at another position, or destination, within the document.

Destinations may also be text that is used but should not appear within the document at all. An example of a destination is the \footnote group, where the footnote text follows the control word.

Page breaks cannot occur in destination text. Destination control words and their following text must be enclosed in braces. No other control words or text may appear within the destination group.

Destinations added after the RTF Specification published in the March 1987 *Microsoft Systems Journal* may be preceded by the control symbol *. This control symbol identifies destinations whose related text should be ignored if the RTF reader does not recognize the destination. (RTF writers should follow the convention of using this control symbol when adding new destinations or groups.)

Destinations whose related text should be inserted into the document even if the RTF reader does not recognize the destination should not use *. All destinations that were not included in the March 1987 revision of the RTF Specification are shown with * as part of the control word.

Formatting specified within a group affects only the text within that group. Generally, text within a group inherits the formatting of the text in the preceding group.

However, Microsoft implementations of RTF assume that the footnote, annotation, header, and footer groups (described later in this chapter) do not inherit the formatting of the preceding text. Therefore, to ensure that these groups are always formatted correctly, you should set the formatting within these groups to the

default with the **\sectd, \pard,** and **\plain** control words, and then add any desired formatting.

The control words, control symbols, and braces constitute control information. All other characters in the file are plain text. Here is an example of plain text that does not exist within a group:

```
{\rtf\ansi\deff0{\fonttbl{\f0\froman Tms Rmn;}{\f1\fdecor
Symbol;}{\f2\fswiss Helv;}}{\colortbl;\red0\green0\blue0;
\red0\green0\blue255;\red0\green255\blue255;\red0\green255\
blue0;\red255\green0\blue255;\red255\green0\blue0;\red255\
green255\blue0;\red255\green255\blue255;}{\stylesheet{\fs20
\snext0Normal;}}{\info{\author John Doe}
{\creatim\yr1990\mo7\dy30\hr10\min48}{\version1}{\edmins0}
{\nofpages1}{\nofwords0}{\nofchars0}{\vern8351}}\widoctrl\ftnbj
\sectd\linex0\endnhere \pard\plain \fs20 This is plain text.\par}
```

The phrase "This is plain text" is not part of a group and is treated as document text.

As previously mentioned, the backslash (\) and braces ({ }) have special meaning in RTF. To use these characters as text, precede them with a backslash, as in \\, \{, and \}.

Conventions of an RTF Reader

The reader of an RTF stream is concerned with the following:

- Separating control information from plain text.
- Acting on control information.
- Collecting and properly inserting text into the document, as directed by the current group state.

Acting on control information is designed to be a relatively simple process. Some control information simply contributes special characters to the plain text stream. Other information serves to change the *program state*, which includes properties of the document as a whole, or to change any of a collection of *group states*, which apply to parts of the document.

As previously mentioned, a group state can specify the following:

The *destination*, or part of the document that the plain text is constructing.

- Character-formatting properties, such as bold or italic.
- Paragraph-formatting properties, such as justified or centered.
- Section-formatting properties, such as the number of columns.
- Table-formatting properties, which define the number of cells and dimensions of a table row.

In practice, an RTF reader will evaluate each character it reads in sequence as follows:

- If the character is an opening brace ({), the reader stores its current state on the stack. If the character is a closing brace (}), the reader retrieves the current state from the stack.

- If the character is a backslash (\), the reader collects the control word or control symbol and its parameter, if any, and looks up the control word or control symbol in a table that maps control words to actions. It then carries out the action prescribed in the table. (The possible actions are discussed below.) The read pointer is left before or after a control-word delimiter, as appropriate.

- If the character is anything other than an opening brace ({), closing brace (}), or backslash (\), the reader assumes that the character is plain text and writes the character to the current destination using the current formatting properties.

If the RTF reader cannot find a particular control word or control symbol in the look-up table described above, the control word or control symbol should be ignored. If a control word or control symbol is preceded by an opening brace ({), it is part of a group.

The current state should be saved on the stack, but no state change should occur. When a closing brace (}) is encountered, the current

to correctly interpret RTF written to this syntax. It is worth mentioning again that RTF readers do not have to use all control words, but they

There may, however, be RTF writers that generate RTF that does not conform to this syntax, and as such, RTF readers should be robust enough to handle some minor variations. Nonetheless, if an RTF writer generates RTF conforming to this specification, then any correct RTF reader should be able to interpret it.

Header

The header has the following syntax:

<header>	***\rtf*** <charset> ***\deff***? <fonttbl> <filetbl>? <colortbl>? <stylesheet>? <listtables>? <revtbl>?

Each of the various header tables should appear, if they exist, in the above order. Document properties can occur before and between the header tables. A property must be defined before being referenced. Specifically:

- The style sheet must occur before any style usage.
- The font table must precede any reference to a font.
- The **\deff** keyword must precede any text without an explicit reference to a font, because it specifies the font to use in such cases.

RTF Version

An entire RTF file is considered a group and must be enclosed in braces. The **\rtf*N*** control word must follow the opening brace. The

There may, however, be RTF writers that generate RTF that does not conform to this syntax, and as such, RTF readers should be robust enough to handle some minor variations. Nonetheless, if an RTF writer generates RTF conforming to this specification, then any correct RTF reader should be able to interpret it.

Header

The header has the following syntax:

<header>	**\rtf** <charset> **\deff**? <fonttbl> <filetbl>? <colortbl>? <stylesheet>? <listtables>? <revtbl>?

Each of the various header tables should appear, if they exist, in the above order. Document properties can occur before and between the header tables. A property must be defined before being referenced. Specifically:

- The style sheet must occur before any style usage.
- The font table must precede any reference to a font.
- The **\deff** keyword must precede any text without an explicit reference to a font, because it specifies the font to use in such cases.

RTF Version

An entire RTF file is considered a group and must be enclosed in braces. The **\rtf*N*** control word must follow the opening brace. The

numeric parameter ***N*** identifies the major version of the RTF Specification used. The RTF standard described in this Application Note, although titled as version 1.5, continues to correspond syntactically to RTF Specification version 1. Therefore, the numeric parameter ***N*** for the **\rtf** control word should still be emitted as 1.

Character Set

After specifying the RTF version, you must declare the character set used in this document. The control word for the character set must precede any plain text or any table control words. The RTF Specification currently supports the following character sets and many more.

Control word	**Character set**
\ansi	ANSI (the default)
\mac	Apple Macintosh
\pc	IBM PC
\pca	IBM PC, used by IBM Personal System/2 (not implemented in version 1 of Microsoft Word for OS/2)

Converting a document from Microsoft's Rich Text Format to plain text format will encounter the following constraints.

- Multi-section properties are ignored.
- No content synchronisation.
- Only simple text tabulation is supported.
- Formulas are ignored.
- Manually numbered footnotes are not supported.
- Annotation text is ignored.
- Not all numbering schemes are supported.
- Document management attributes are lost.
- Generation of incomplete generic layout structures cause other parsers to fail.
- New page information loss leads to page breaks not being generated.

REPORT VIEWER

MAIN FEATURES

REPORT VIEWER
MAIN FEATURES

INTRODUCTION

The software is being developed with the following aspects under consideration, which form the basis of generating the code in order. These factors form the overall issues in the underlying project.

- Creating a project
- Creating an application
- Creating a frame
- Creating a dialog
- Creating a panel
- Creating a data module
- Creating a class
- Converting an HTML file
- Overriding methods
- Implementing a Java interface
- Bundling resources
- Specifying an action for the component

Building a Java text editor

This Java application called Report Viewer, which is a simple text editor capable of reading, writing, and editing text files. It is also able to set the text font and color, as well as the background color of the text editing region.

It steps you through handling events for commonly used components and tasks, like menu items, ButtonBar, TextArea, and system events.

It contains other menu for File|events Edit|**events,** Search|events, Help|events etc., for further processing the file that is being created or that has opened an existing file.

Overview of menu components and terminology

Illustrates the general menu terminology used in report viewer. Also discusses the two types of menu components used in Java programs: MenuBar and PopupMenu.

The basic parts of a menu are referred to using the following terms:
The menu bar displays at the top of a frame and is composed of menus containing individual menu items.
A menu item is an individual element on a menu.
Menu items can have attributes such as being disabled (gray) when not allowed, or checkable so their selection state can be toggled.
A menu is a list of menu items.
A submenu is a nested menu accessed by clicking on an arrow to the right of a menu item.

A keyboard shortcut is displayed to the right of the menu item, and may be specific to a particular editor interface. For example, Ctrl-X is used to indicate Edit|Cut in many editors.

The separator bar helps to visually group related items together.

Types of menu components

There are two types of menu components on the Component : a MenuBar and a PopupMenu (local
context menu).

A MenuBar is attached to the main Frame, and appears at the top of the application.
A PopupMenu appears when the user right-clicks in the TextArea.
Popup menus do not have menu bars.
Both kinds of controls contain menu items which can be inspected by the User.
The first MenuBar control dropped onto the textarea is considered the current MenuBar.

However, you can create more than one MenuBar for an application, and they are displayed in the Inspector in the frame's MenuBar property. To change which menu is the current MenuBar, just select a menu from the MenuBar property drop-down list.

Note: Menu components are only visible at run time in
the MenuBar.

Creating a menu

Explains how to open the Menu and create a new menu. Tells you how to add, insert, and delete items, enter separator bars and shortcuts, and disable (dim) a menu item or make it checkable.

Specifying keyboard shortcuts

Keyboard shortcuts enable the user to perform an action without accessing the menu directly by typing in a shortcut key combination. For example, a commonly used shortcut for File|Save is Ctrl+S.

To specify a keyboard shortcut for a menu item,
Select the menu item in the Menu.

In the Inspector, select the shortcut property and enter a value or choose a key combination from the drop-down list. This list is only a subset of the valid combinations you can type in.

The shortcut appears next to the menu item at runtime.

Disabling (dimming) menu items

You can prevent users from accessing certain menu commands based on a particular state of the current program conditions, without removing the command from the menu. For example, if no text is

currently selected in a document, the Undo|Redo items on the Edit menu appear dimmed.

Disabling a menu item is done with the Enabled property for the menu item. The default state of the menu item is True, and this changes when some event occurs, like the selection of text.

Creating checkable menu items

To make a menu item checkable, first you need to change the menu item from a regular MenuItem component to a CheckboxMenuItem. A CheckboxMenuItem has a State property (boolean) that determines if the associated event, or behavior, should be executed.

A checked menu item has its State property set to true.
An unchecked menu item has its State property set to false.
To change a regular menu item to a CheckboxMenuItem, select the menu item and click the Check button on the menu item for example Word-wrap menu item is a checkable menu item.

Inserting a separator bar

Separator bars insert a line between menu items. You can use separator bars to indicate groupings within the menu list, or simply to provide a visual break in a list.

The separator bar is inserted above the selected menu item.
Separator bar is used for distinguishing color menu items, font menu items, print menu, search through pages, search through key words, etc.,

Moving among menu items

In the Menu you can move through menus and menu items simply by dragging them with the mouse. When you move through a menu or a submenu, any items contained in its position respond to the respective event handling.

You can move through the following:

- Menus along the menu bar.
- Menu items within a menu.
- Menu items to other menus.
- Submenu items to other menus.

Creating submenus

Explains how to create submenus (nested menus) accessible from an arrow to the right of a menu item.

Many applications have menus containing drop-down lists that appear next to a menu item to provide additional, related commands. Such lists are indicated by an arrow to the right of the menu item. It

supports as many levels of such nested menus, or submenus, as you want to build into your menu.

Organizing your menu structure this way can save vertical screen space. However, for optimal design purposes you probably want to use no more than two or three menu levels in the frame.

Attaching code to menu events

Demonstrates how to hook up menu items to events.

A menu item has only one event: actionPerformed. Code that you add to the actionPerformed event for a menu item is executed whenever the user chooses that menu item or uses its accelerator or shortcut keys.

To add code to a menu item's actionPerformed event

- Open the frame containing the menubar, you can open a file that already contains a file or create new one.
- Select a menu item in the MenuBar.
- Click the actionPerformed value field to create an event-handling method stub in the source code with a default name.
- The cursor is positioned in the body of the newly created actionPerformed event-handling method, ready for your action on the content. Enter or edit the code you want to have executed when the user clicks this Menu command.

Edit the code for the actionPerformed event to look something like this:

Inside the body of the actionPerformed method type the following:

```
VoidConVactionPerformed(java.awt.event.Action
Event e) {
        StringactionCommand                    =
e.getActionCommand();
        if (actionCommand.equals("Open")) {
          fileOpen();
        }
        else if (actionCommand.equals("Save"))
{
          saveFile();
        }
        else                                   if
(actionCommand.equals("About")) {
          helpAbout();
        }
      }
```

Add more else if statements for all additional labels in the viewer. Note that in this example we are making calls to three methods:fileOpen(), saveFile() and helpAbout().

In a real program, you will typically need to add several lines of custom code in the event-handling methods. In this example, you might want to extract the file name the user entered from the Open and use it to open or manipulate a file.

EDITOR OPTIONS

EDITOR OPTIONS

Displaying the Editor

To display the Editor, first navigate to the text file you want to view or modify. Then click the Source tab named File. The file is displayed in the Content pane or Frame, ready for editing.

Moving around in text files

Ways to move around in text

Action	Feature
Select variables and methods to navigate to	The menu items are displayed in the menubar holding separate menu for all funtions
Open File menu	To process an existing file|new and after editing to save or to print the content present
Edit option menu	To perform undo|redo, formatting the text and to delete or paste a particular part of the text
A option menu	To set the page length and to move the cursor to the particular page and to word-wrap the text

Search menu	To move around the content of the text
Go to a specific line number	Search | Go To Line Number
Find and replace text	Search|Find and Search|Replace
Help menu	To find the available help option like shortcut keys and other functionality

Moving through the structure of a file

To move through a text file, you can:

- Method names and variable names (for Java files).
- Headings and links in the Structure pane (for HTML files).

Reading and writing file content

You can open an existing file or create a new file using either menu commands or keystrokes. The Editor enables you to perform the following standard File commands:

- New, Open, Close
- Save, SaveAs
- Display status, print, exit

You use the following commands for performing these commands:

Command	Description
File|New	Creates a blank text area to write in any content new to the file
File|Open	Inserts text from the existing file from any directory that can be chosen from the open dialog box
File|Close	Closes the file that was opened currently by giving a warning to save or not.
File|Save	Save the file in its own filename
File|SaveAs	Save the content in a different filename | directory
File|Print	To take a hardcopy of the file being currently opened
File|Display Status	Description of the currently opened file status is displayed
File|Exit	Close the entire application

Editing text

You can edit text using either menu commands or keystrokes. The Editor enables you to perform the following standard editing commands:

- Undo and redo
- Format | Font size, Font Style, Color(Background,Foreground)
- Cut, copy, paste and delete

Cutting, copying, pasting, and deleting text

You use the following commands for cutting, copying, pasting and deleting text:

Command	Description
Edit|Undo	Remove the last action performed
Edit|Redo	Replace the last action being Undo is done
Edit|Font	The size, Style(Bold,Italic,Plain) are being changed to the user's wish
Edit|Color	The foreground color and the background color under which the text is displayed can be changed for viewing the report
Edit|Cut	Cuts selected text and moves it to the clipboard.
Edit|Paste	Inserts text in the clipboard to a new location.
Edit|Copy	Copies the selected text to the clipboard.

Undoing and redoing editing actions

You can use the Undo command to undo your last keystrokes, action or deletion. Undo multiple successive actions by continuing to use the Undo command. You can specify a limit to the number of previous actions you can undo on the Editor page of the of the Environment Options dialog box. Undo enables you to reverse entire block operations.

To undo your last keystrokes, action or deletion,

1. Choose Edit|Undo, or
2. Press Ctrl+Z.

To reverse an undo,

- Choose Edit|Redo, or
- Press Shift+Ctrl+Z.

Setting the Page-length | Option menu

To move around the content of the file page wise. We have to set the page-length to set the cursor position to the respective page value and to view the content in the word-wrapped text display, the following commands are performed:

Command	Description
Option|Page-length	A dialog box is opened to enter a numeric value in order to set the page length
File|Goto Page	Again in a dialog box that is opened place a numeric value in order to move the cursor position to set.
File|Word-wrap	The text that is beyond the set columns is being cut and placed in the new line

Search through the file

The Editor enables you to perform the following standard search commands:

- Find, Replace
- Goto Line, first,last
- Previous page, next page

You use the following commands for performing these commands:

Command	**Description**
Search|Find	The text of the keyword is being highlighted in sequence from the top
Search| Replace	The text is being replaced with the find keyword again in sequence of the file being parsed
Search| Goto line	The cursor is set to that caret position being specified numeric value
Search| first	The cursor is placed in the first line of the text
Search| last	The cursor is placed in the first line of the text
Search|Previous page	From the set caret position the cursor is moved to a page before
Search|Next page	From the set caret position the cursor is moved to a page after

Finding and replacing text

The Editor lets you specify the full string you want to find before searching, or you can search for text as you type by using the Editor's incremental search function.

You can search for literal text, or you can use regular expressions to find characters or words.

Finding specific text

To find specific text,

1. Choose Search|Find or press Ctrl+F.
2. The Find Text dialog box is displayed.
3. Specify text you want to locate.
4. Set options that affect the search.
5. Click OK.

Find Text dialog

Replacing text

To replace text,

1. Select Search|Replace or press Ctrl+R.
   The Replace Text dialog is displayed.
2. Specify text you want to search for and then replace with other text.
3. Click OK.

Replace Text dialog

Going to a specific line number

You can go to a specific line number in your text by using the Go To Line Number dialog box, which prompts you for the line number to go to. The current line and character number where your cursor is placed appear in the dialog box when it opens.

To move the cursor to a specific line number in your text,

1. Choose Search|Go to Line Number.
   The Go To Line Number dialog is displayed.
2. Enter the number of the source code line you wish to go to in the Enter New Line Number list box,.
3. Click OK.

The help command consists of information shortcut keys of the entire editing file.

These are the various editor options that are being performed over the report viewer generated. It could access all types of files including the converted files.

IMPLEMENTATION ISSUES

IMPLEMENTATION ISSUES

Overview-System implementations

Implementation is an activity or process of bringing a developed system into operational use and turning it over to the user. Implementation is the key stage in achieving a successful new system, because usually it involves a lot of upheaval in the user side. Normally during the application implementation the users are given training on the usage of the application, to satisfy their core needs.

The users after opening the editor are provided with all options on the menu which on clicking with the mouse the event of the particular action takes place automatically. The user could also use the shortcut keys which he could get the information directly from the menu list or from the help menu.

Separate class files have been created for each action which the user need not worry about. Any type of user working in different system environment can make use of the software as it is developed in Java which is platform independent. So, irrespective of the operating system, the user can make use of the software without any alteration.

The whole software is developed using the AWT(Abstract Window Toolkit) concepts in Java. Mainly, because when it is to

be viewed as a component of reusability it can be bean converted. It forms as a separate component, whereas other features of Java like Swing is not supported by Java Beans.

The files in the html or rtf are format converted in normal text format and are being displayed in the frame of textarea for user's access. He could view such files in the required fonts. The user could search through the files to any part of the text.

System Testing

It proves to be a very user-friendly environment and the users feel very easy to operate and also satisfied with the quick transfer of messages among other formats.

The editor options provides more flexibility to the user and it is more user-friendly. The features in the project proved to be more helpful and useful in viewing the files. The testing of the project is done and proved to be successful. The actions buttons developed as menu-event based system help in easier access of the actions.

UPGRADATION

UPGRADATION

Features available

Format conversion and report viewer's important feature is to convert the .html | .htm and .rtf files to plain text.
It has the capabilities of editing those said text and to print those file by the available format into various fonts.
The converted files are stored in separate text file for faster access later for viewing the report.
The report generated could provide options to move around various positions of the text loaded.

Future Enhancement

This software could be extended by intrepreting the tags and control words present in the formatted file which need to be converted.

Codings could be extended to bring down the images and styles present in the original format.

Editors options can be increased to do more number of undo/redo capabilities and change the scroll-speed, freeze the titles for to display every page that is being viewed.

CONCLUSIONS

CONCLUSIONS

The converters can be used to import and export documents between the different formats. The software, as it is developed in Java is made to support multiple operating environments like UNIX, DOS, windows, macintosh, etc., .

This product is efficient, robust and provides richness enabling the user to communicate in a user-friendly manner.

The software finds applications in various fields for MIS and EIS people for a very good interaction between various level of management and easier to use report viewer.

The above testing has shown, however, that this may result in some loss or changes of information. Some of these points have been highlighted through the testing of only a number of documents.

I conclude that this is the first step toward developing a 'platform independent' report viewer for format conversion for just a html and rtf format files and still complex formats like doc. format and other formats conversions can be included.

BIBLIOGRAPHY

BIBLIOGRAPHY

Reference:

1. Handley, M and J. Crowcroft, *The World Wide Web*, UCL Press, 1995.
2. Patrick Naughton and Herbert Schildt, *The Complete Reference Java^{tm}2* (Third Edition)
3. Joseph Weber, *Special Edition Using Java^{tm}2 Platform*
4. Microsoft WORD User's Guide Microsoft Corporation

The website references:

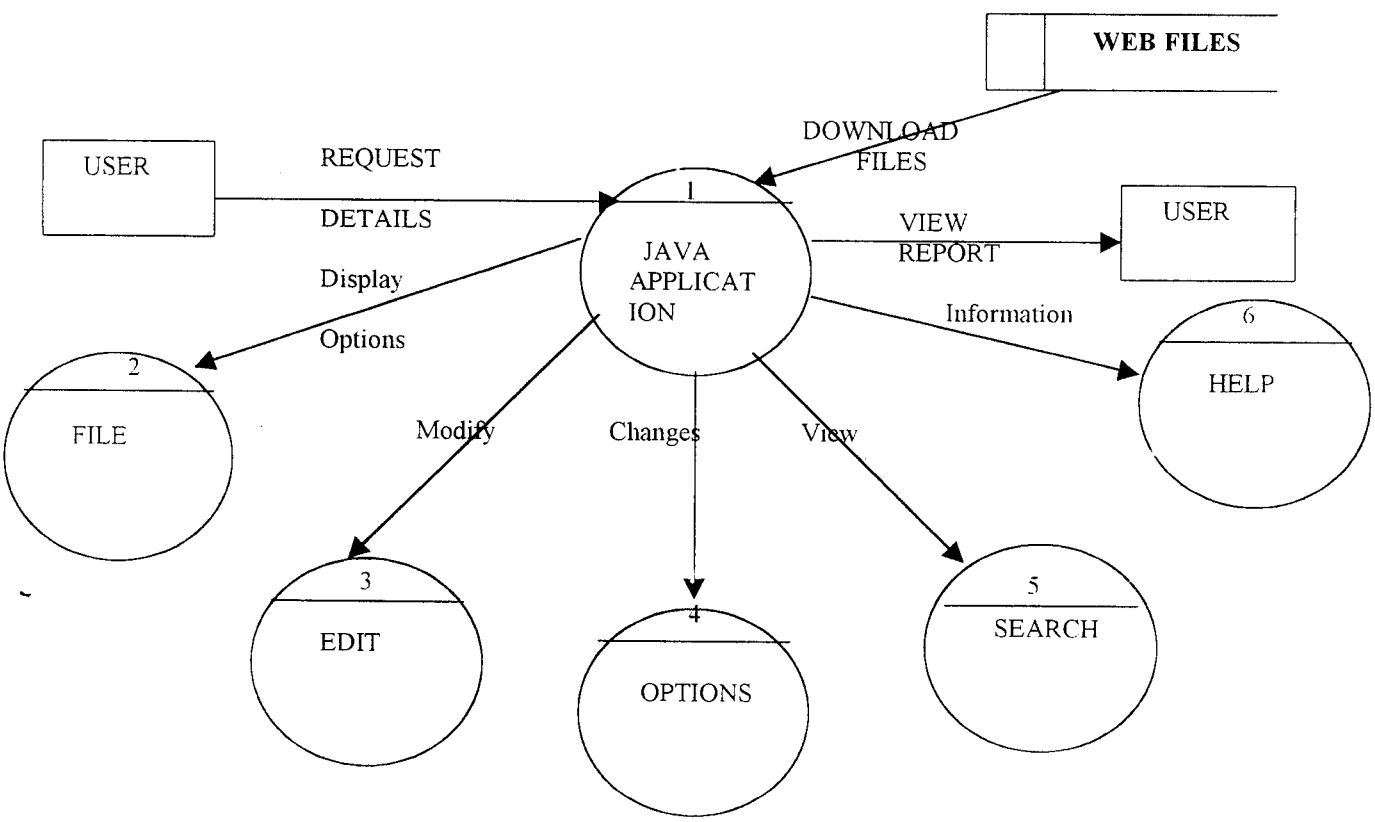
1. An Introduction to latex , http://www.tex.ac.uk/CTAN/latex/intro.html
2. Rich Text Format (RTF) Specification, http://www.w3.org/hypertext/DataSources/RichTextFormat/RTF.txt
3. Gartland, A. and Houghton, D.: *Tool survey report in the context of the DMU-JEDI Project , DMU-JEDI D1* http://jedi.dmu.ac.uk/JEDI/d1/ 1995.
4. HyperText Mark-up Language Specification Version 3.0, http://www.w3.org/hypertext/WWW/MarkUp/html3/CoverPage.html

APPENDIX

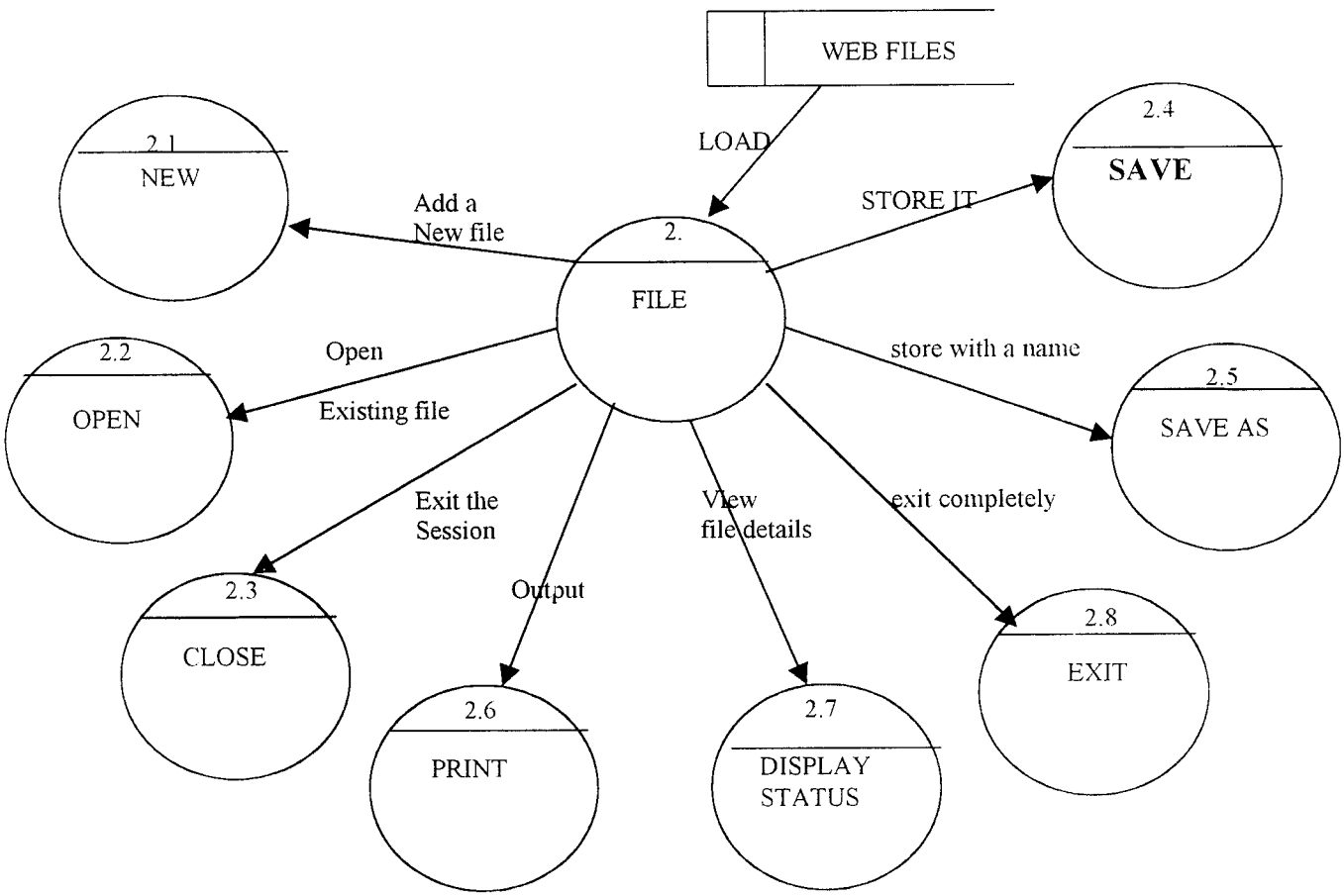
FORMAT CONVERSIONS AND REPORT VIEWER

DATAFLOW DIAGRAM

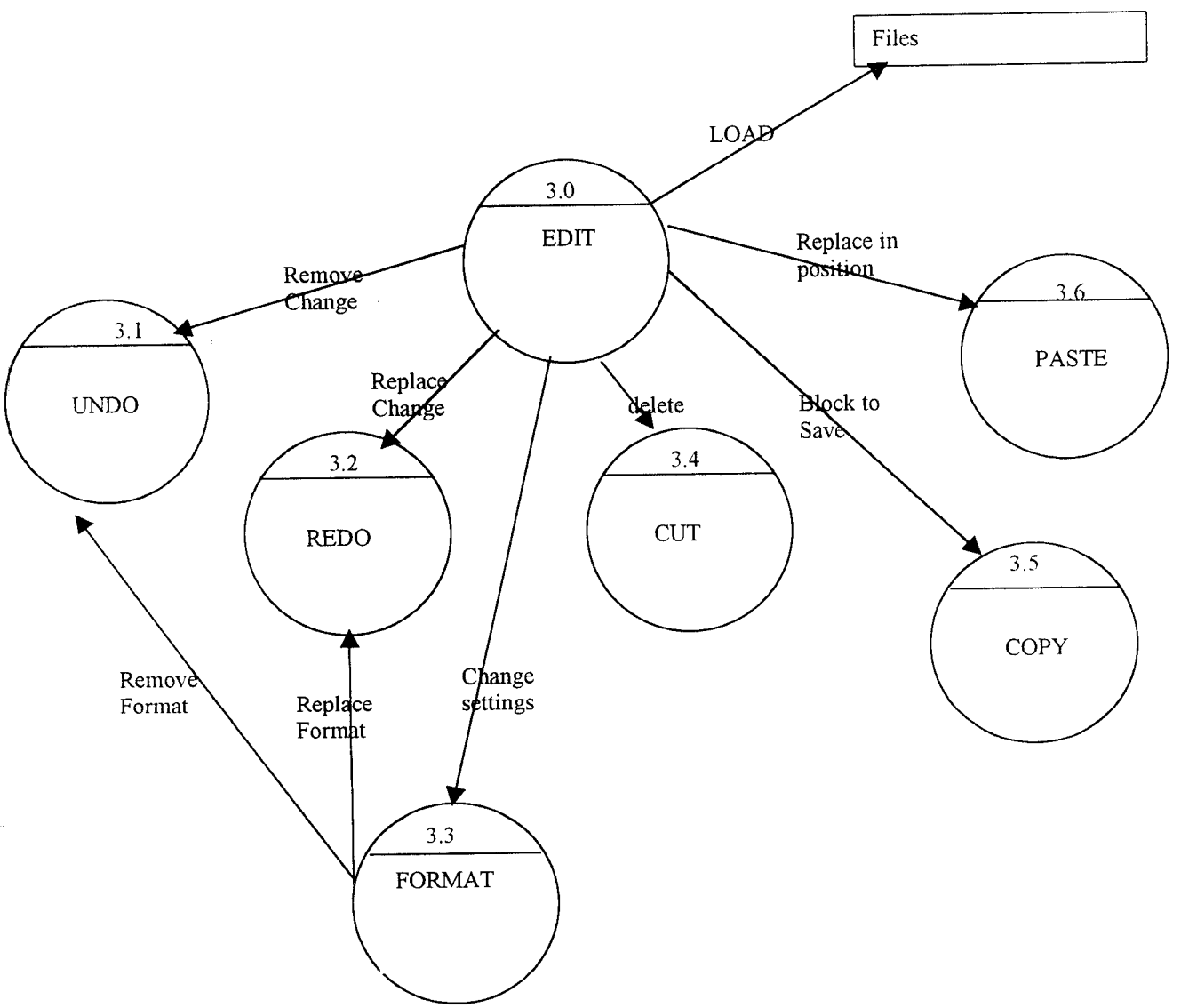
General view of the whole structure



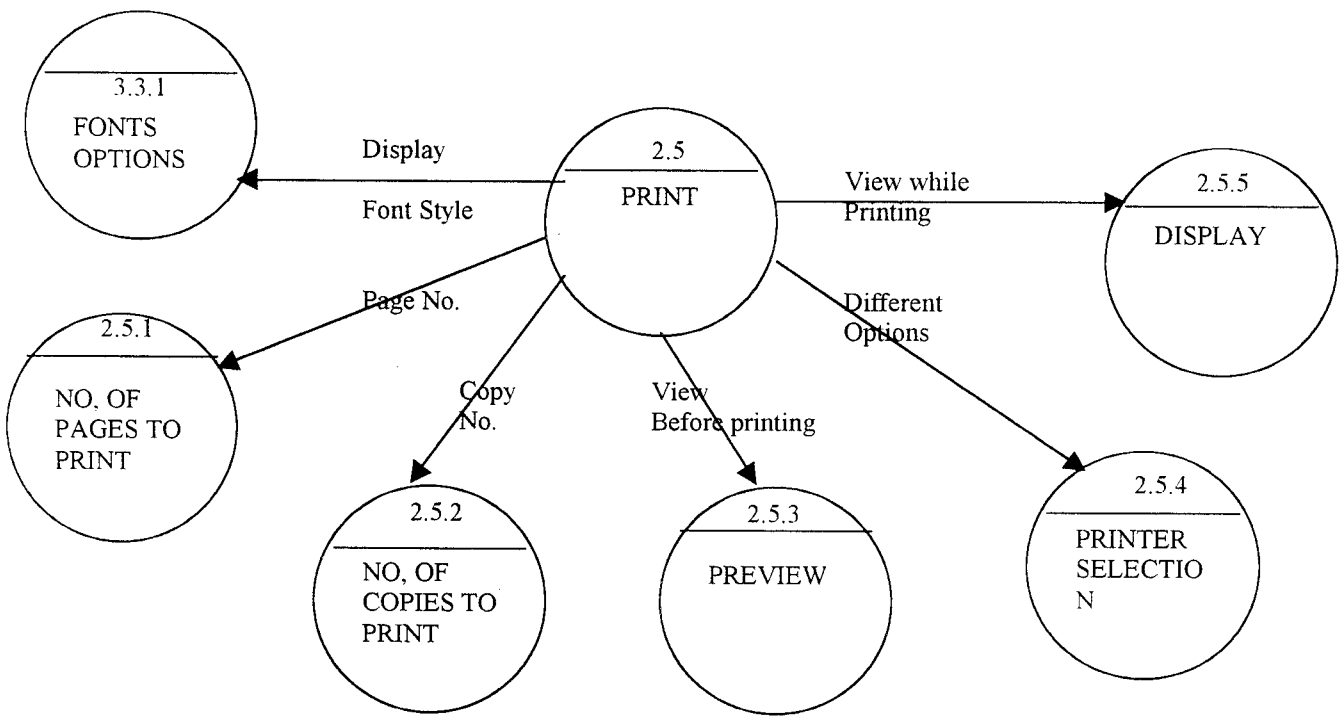
The File Menu design



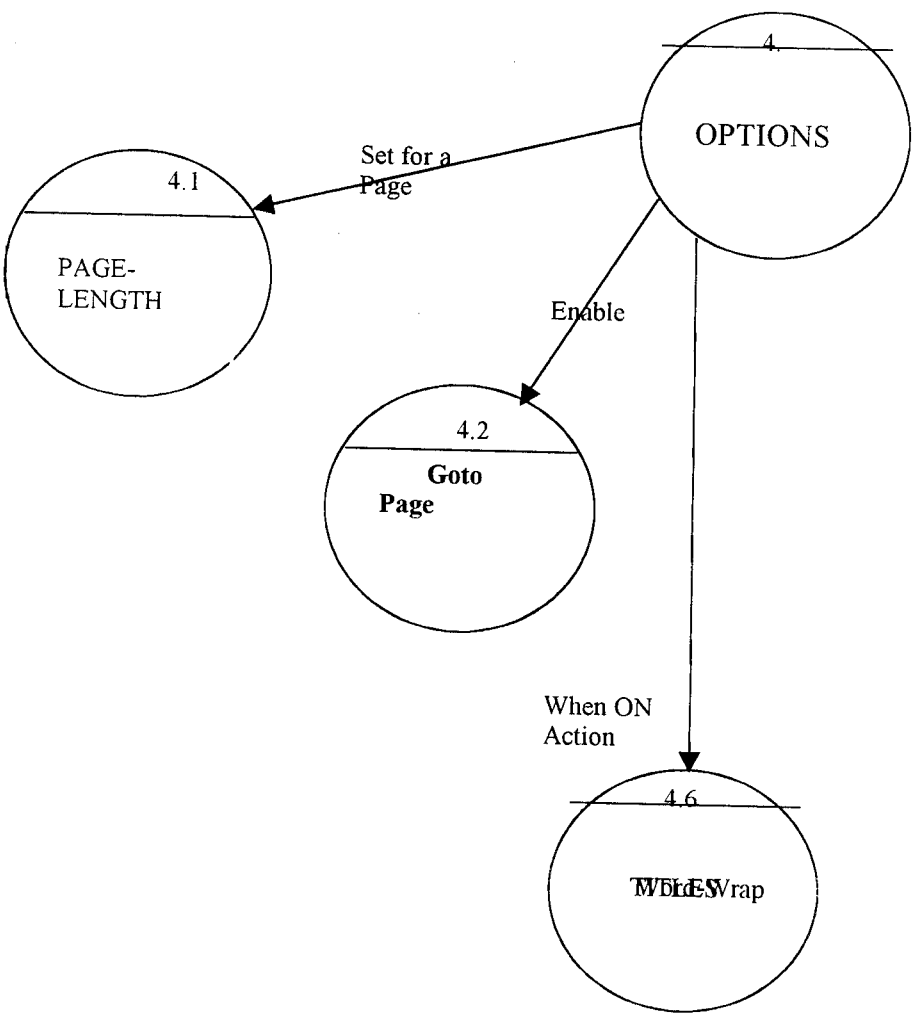
The edit menu design



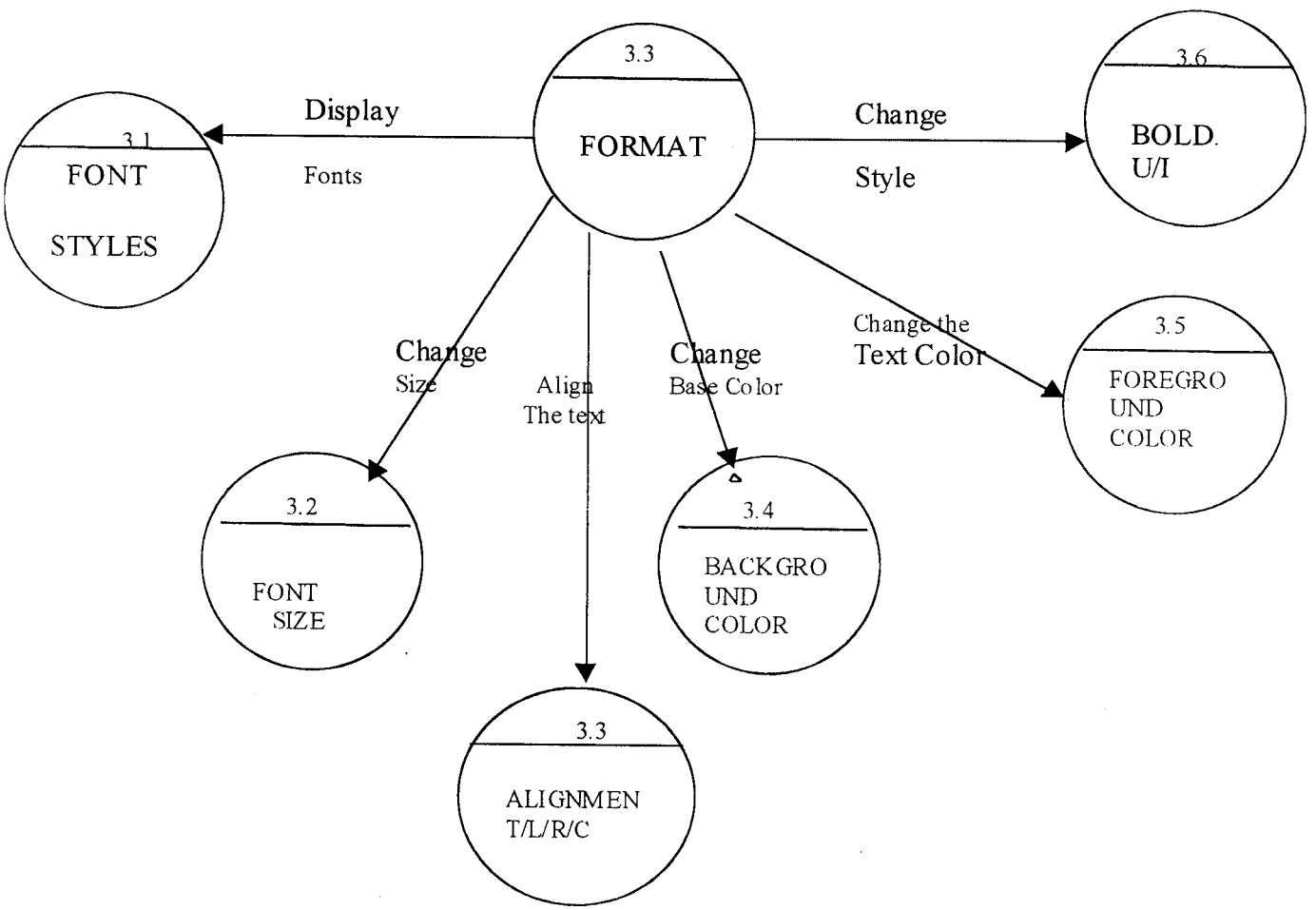
The Print menu design



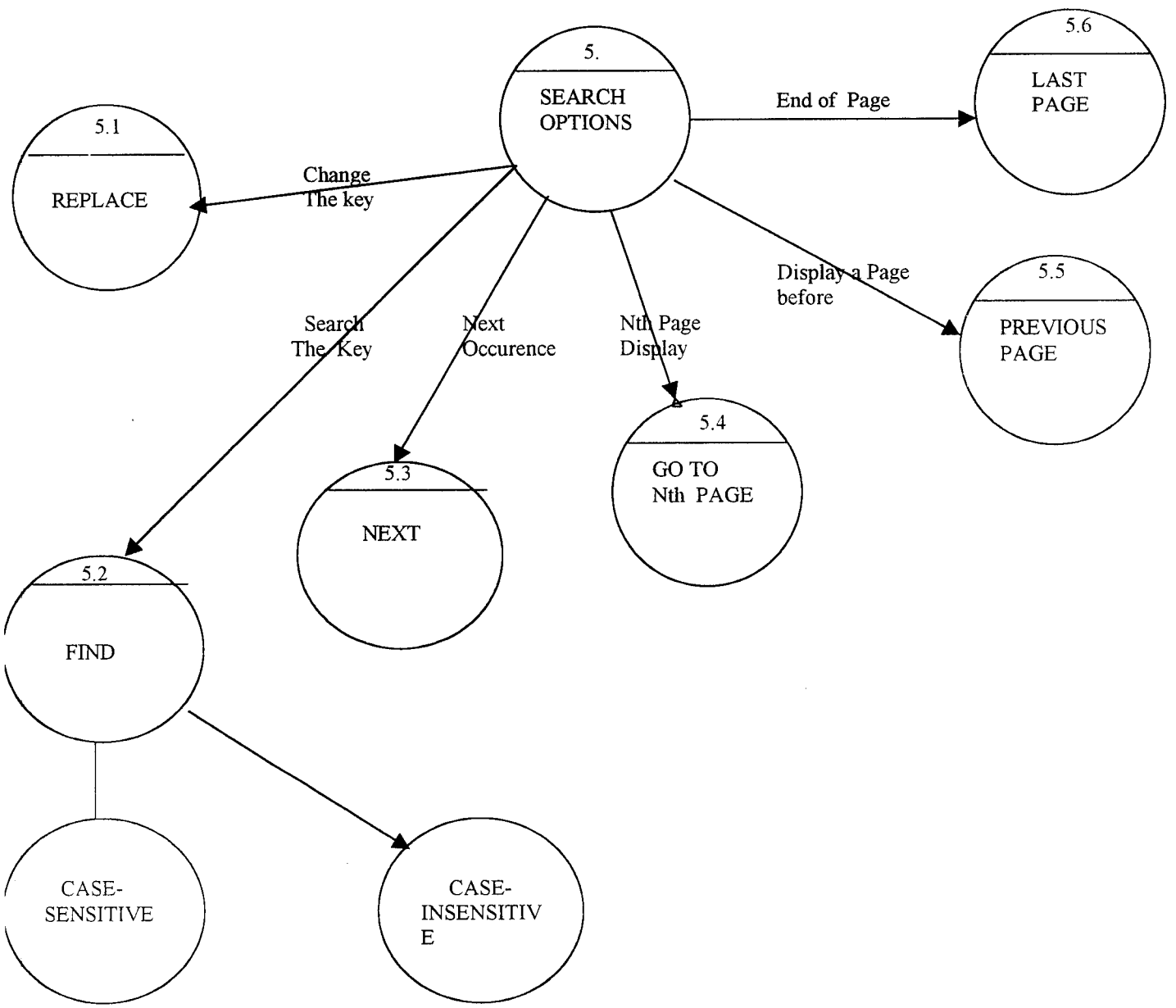
The Option menu design



Different Font Styles

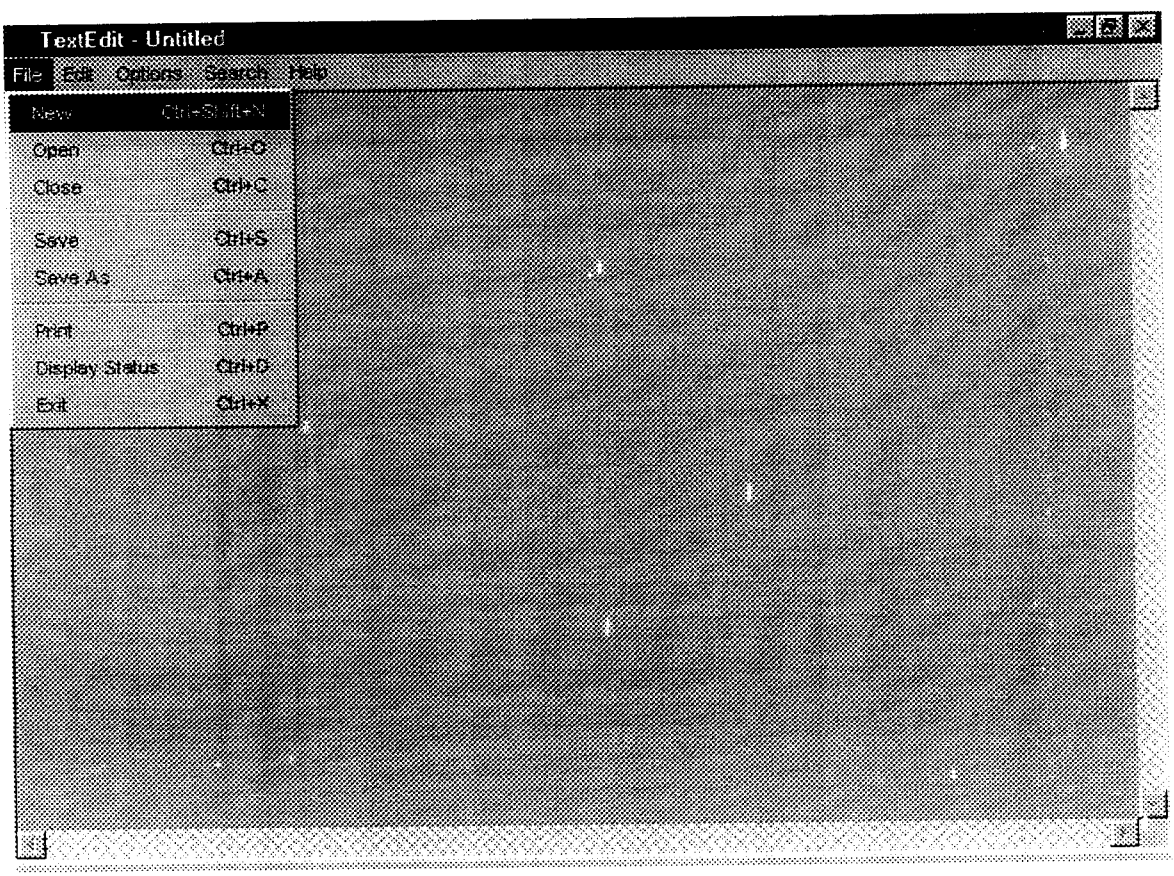


The Search menu design

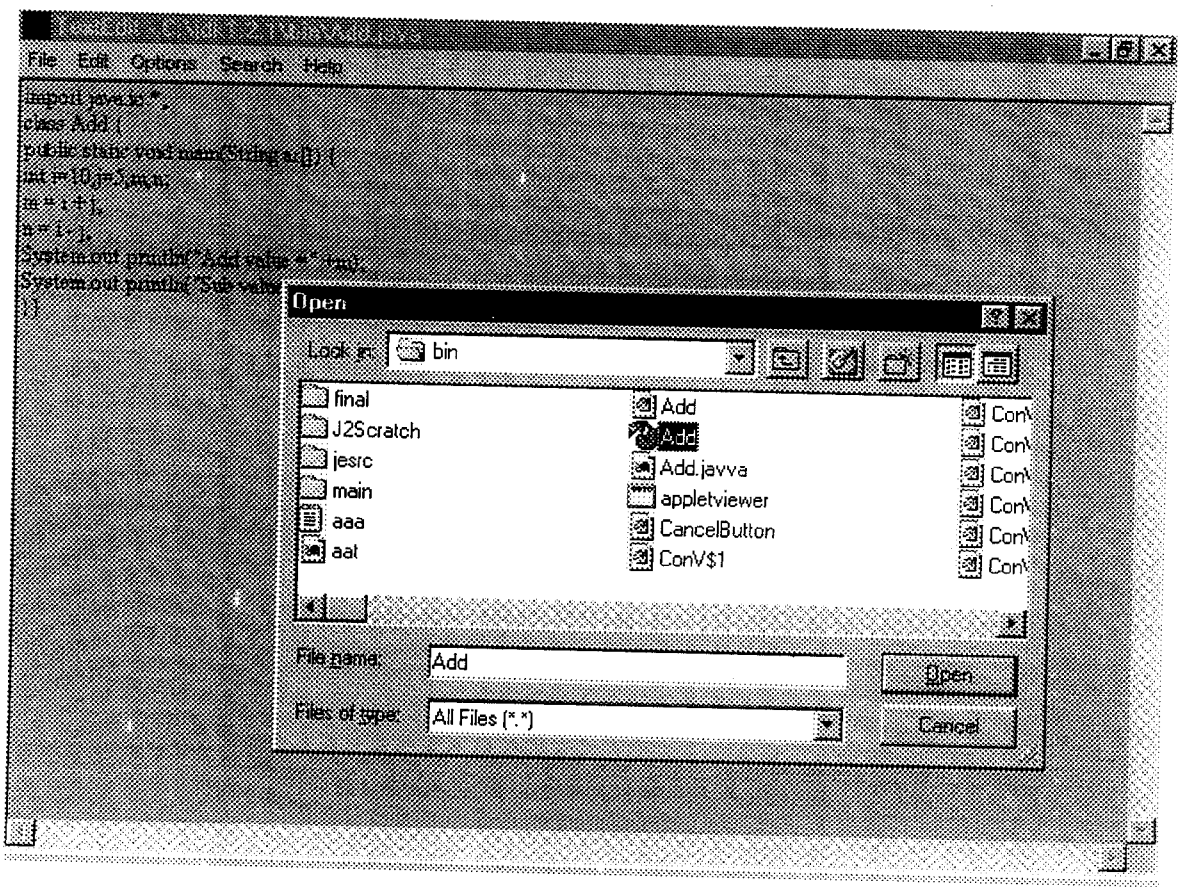


OUTPUT FORMS

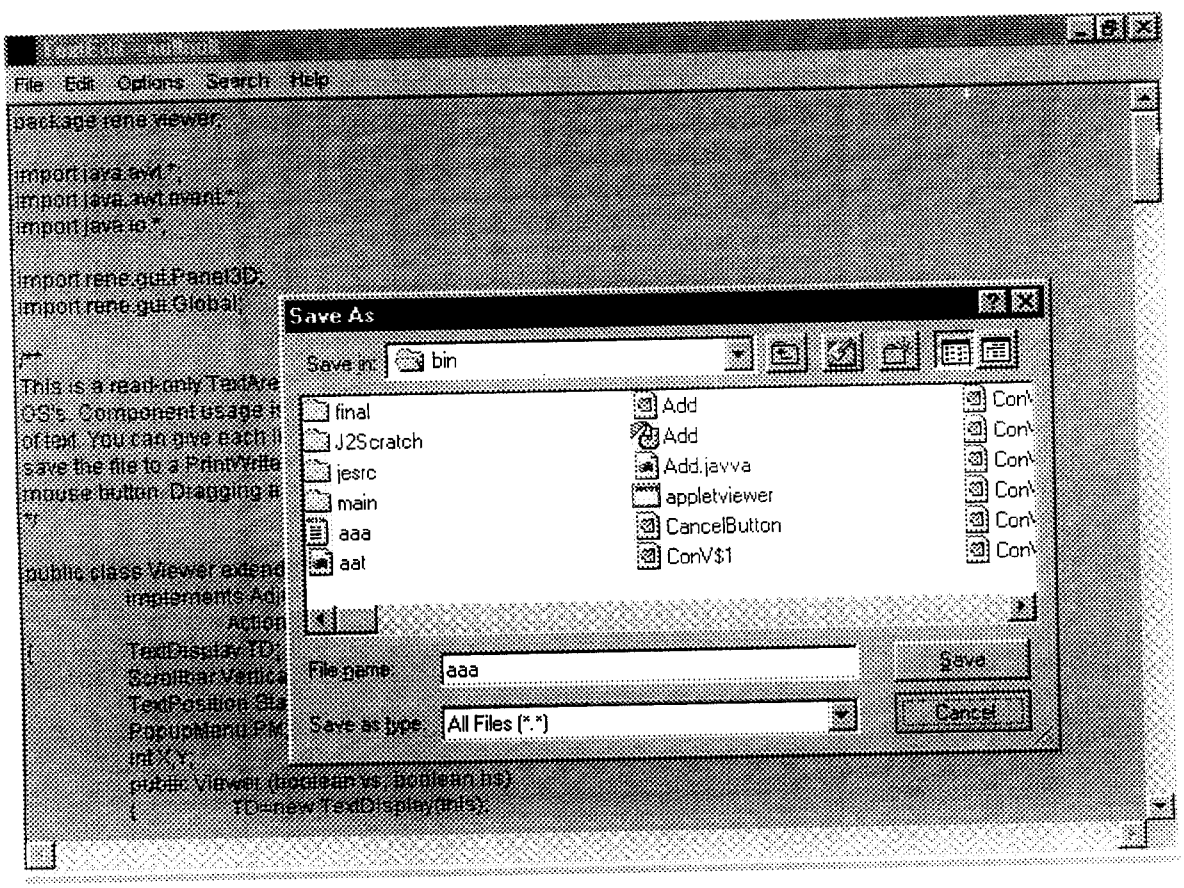
Menu Designs for report viewer for file processing



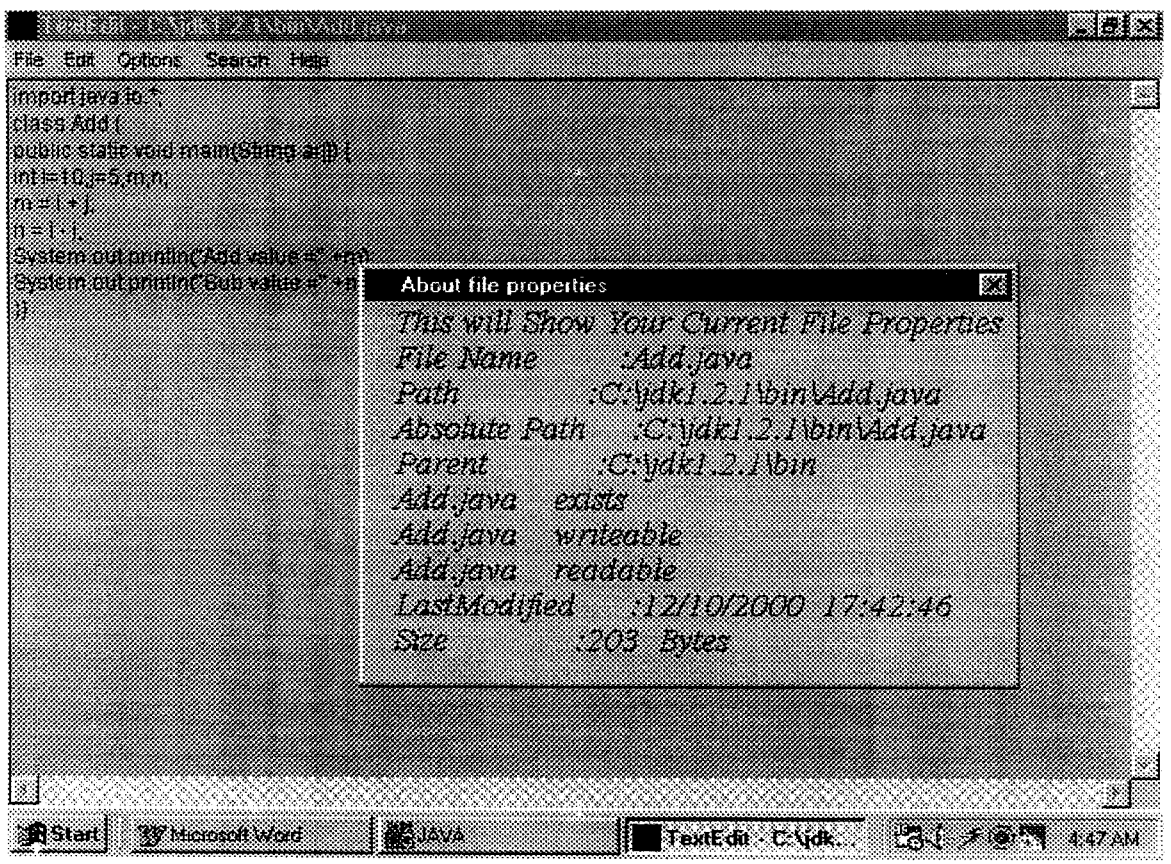
To open an existing file from open dialog box



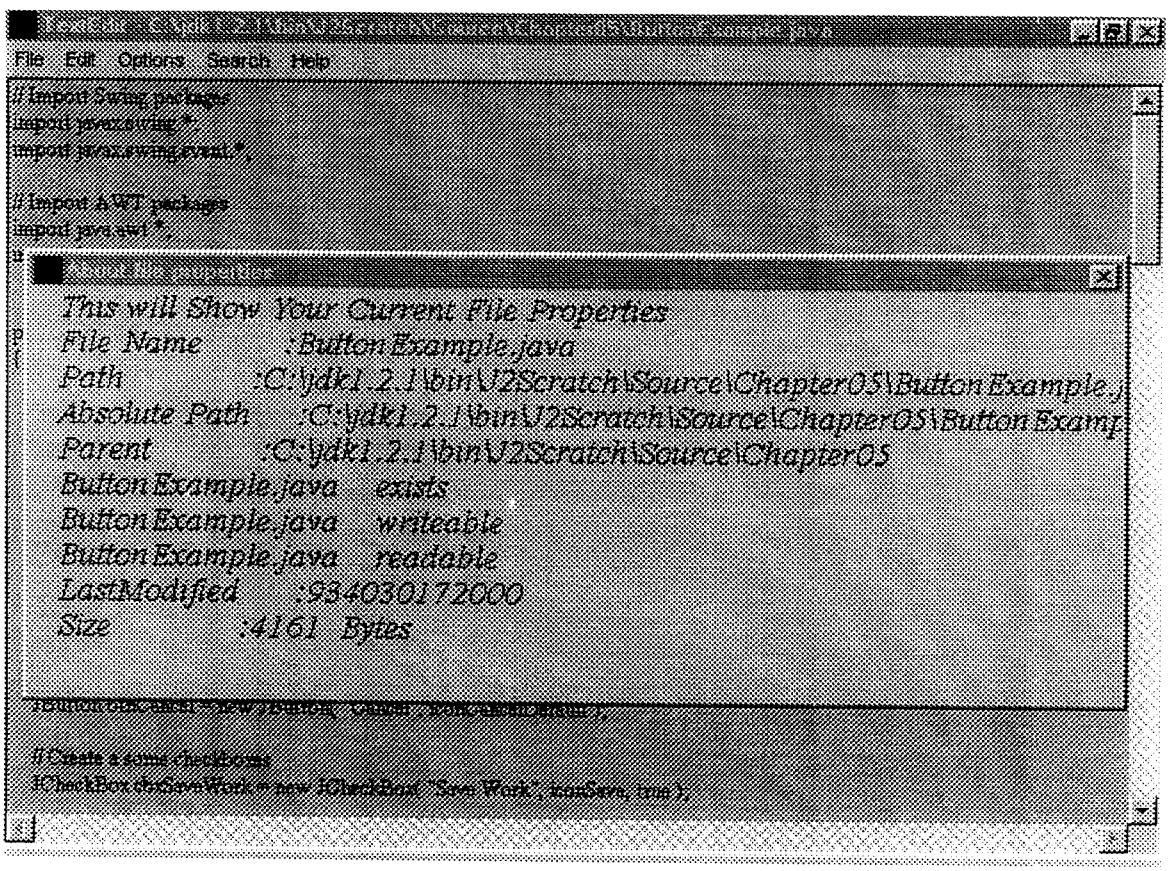
To save a null file use a Save Dialog to be saved in any directory



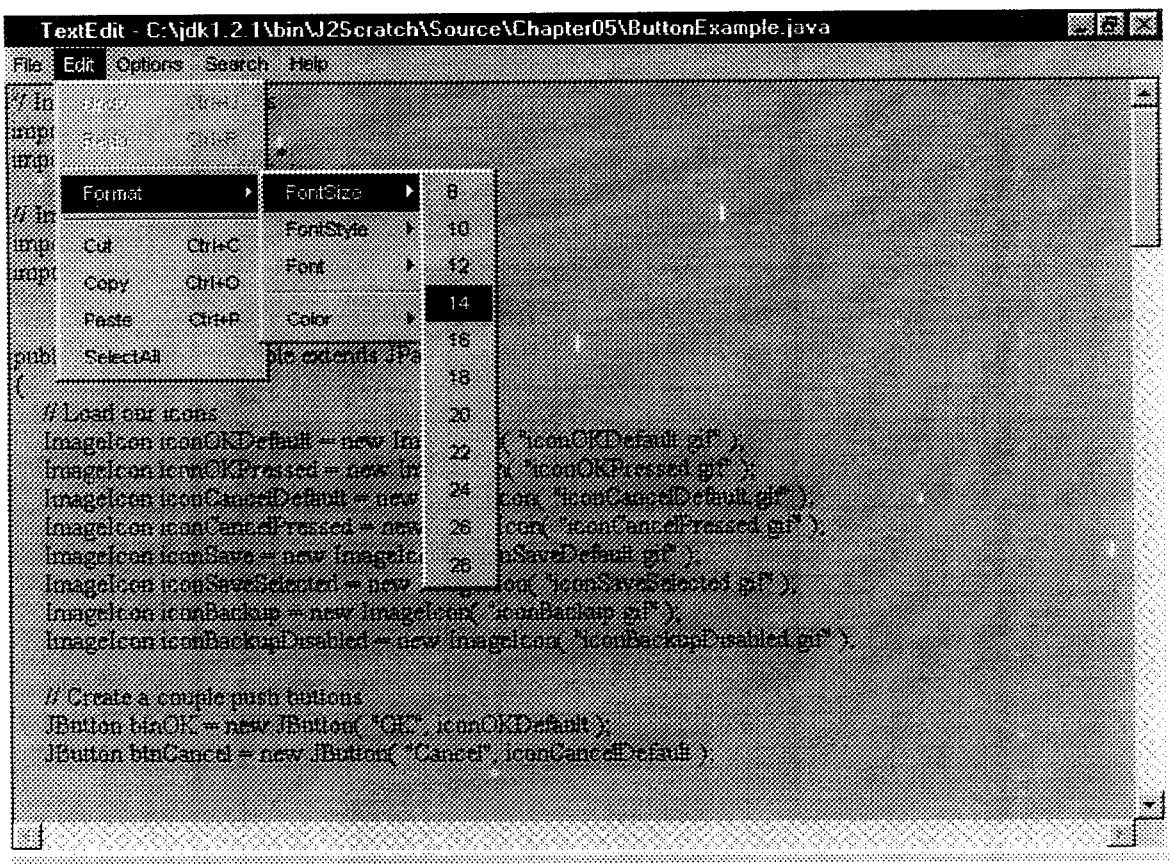
The Properties of the file currently opened using canvas



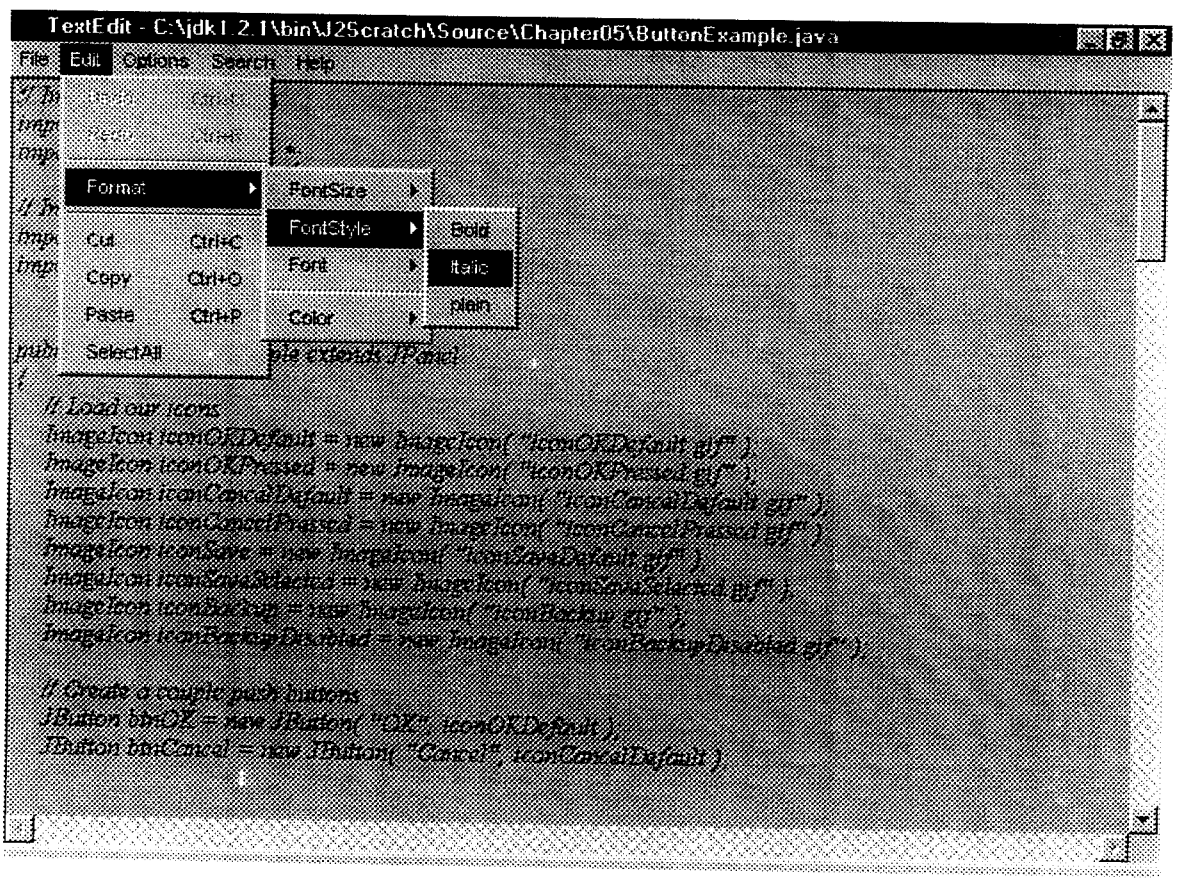
The Properties of a file in different directory opened



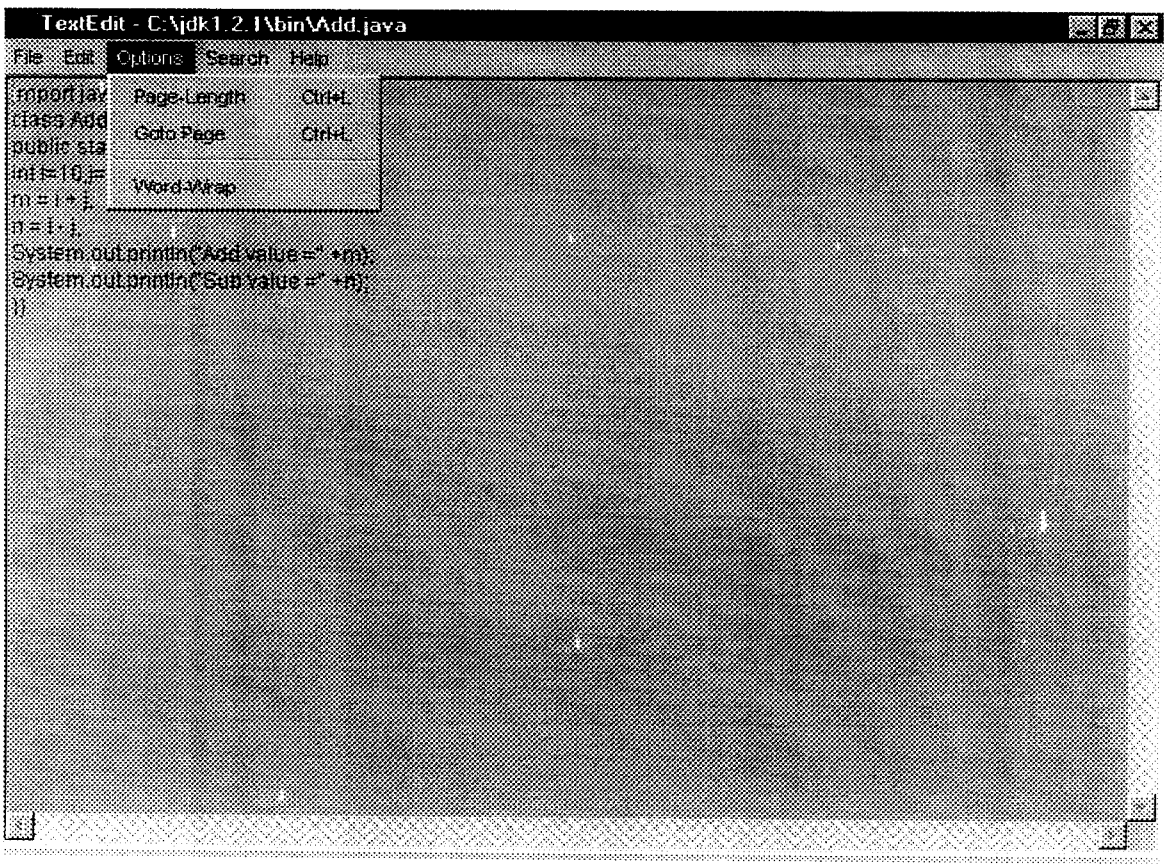
The edit menubar and the menuitems listed in the design
The change of Font size is indicated



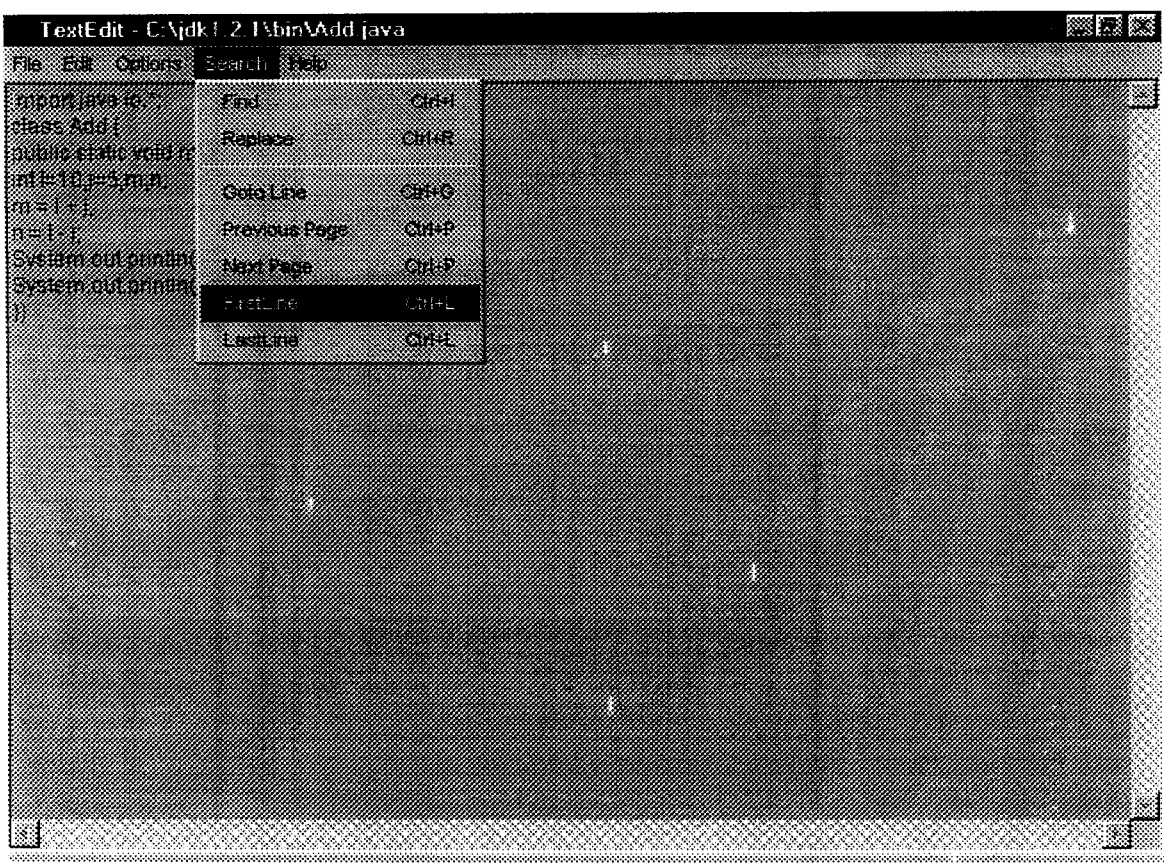
The change in the Font Style



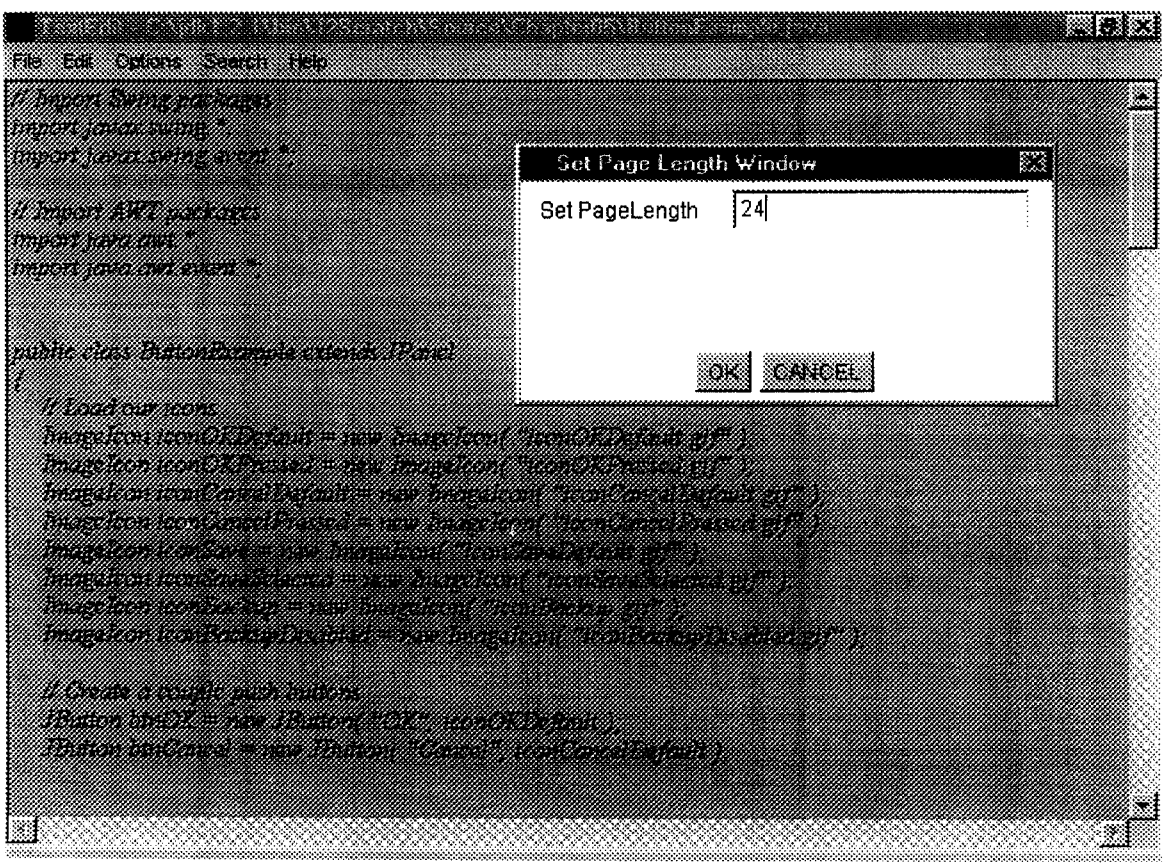
The Options Menubar and its contents design menu



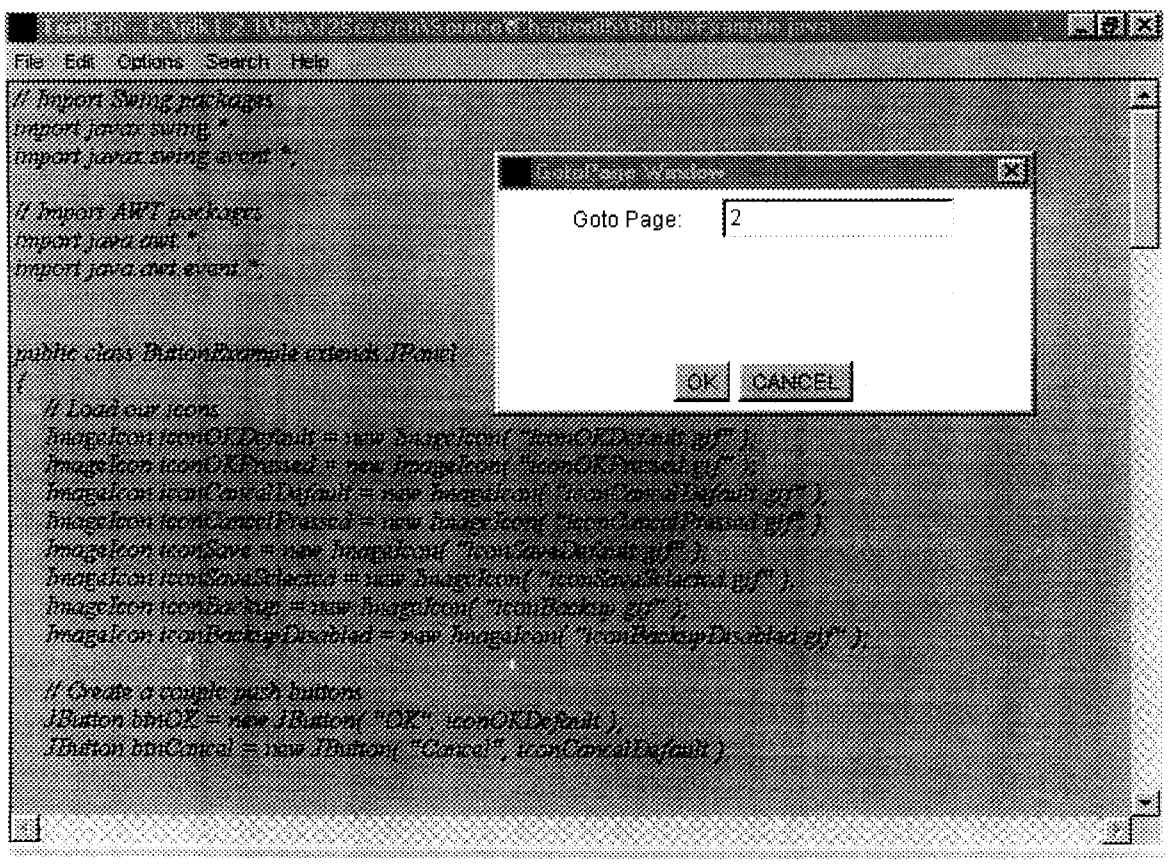
To search around the file through various means



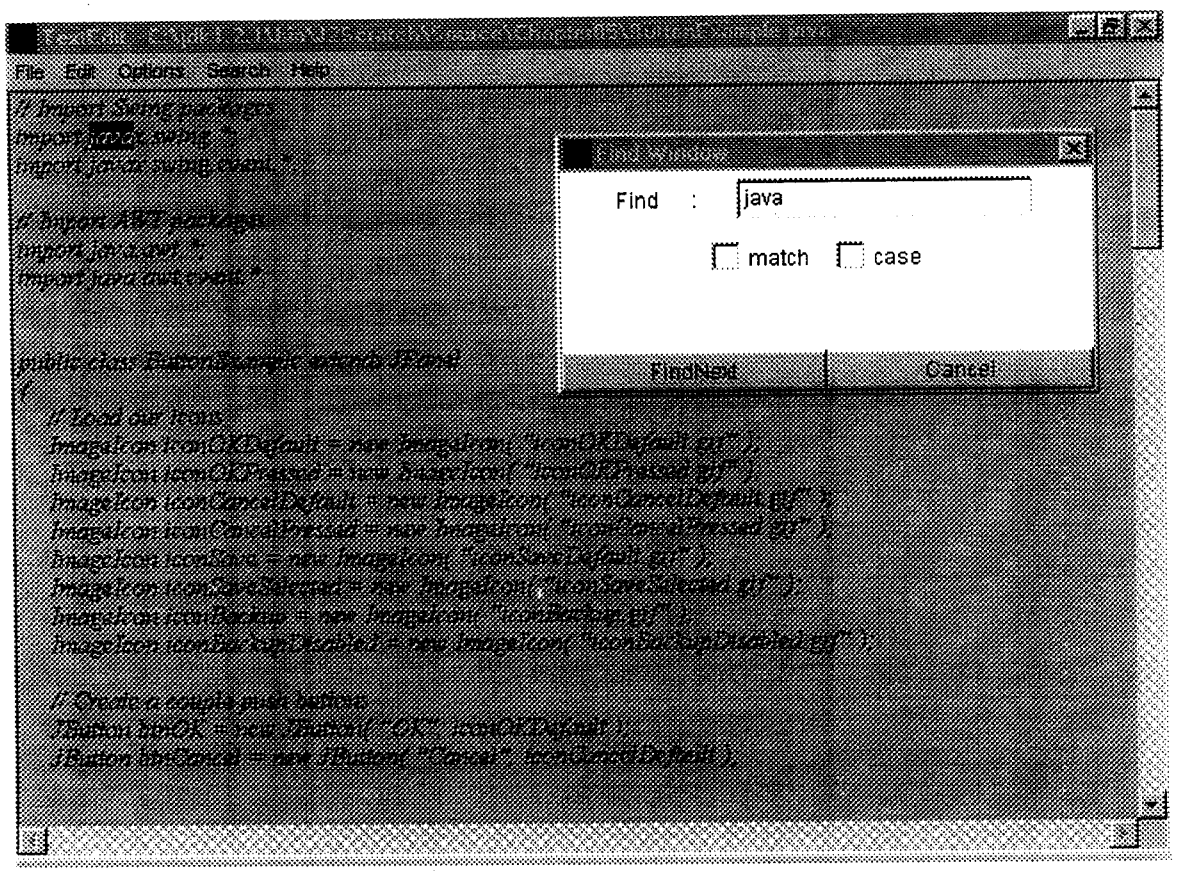
To set the Page Length for searching through the file pages



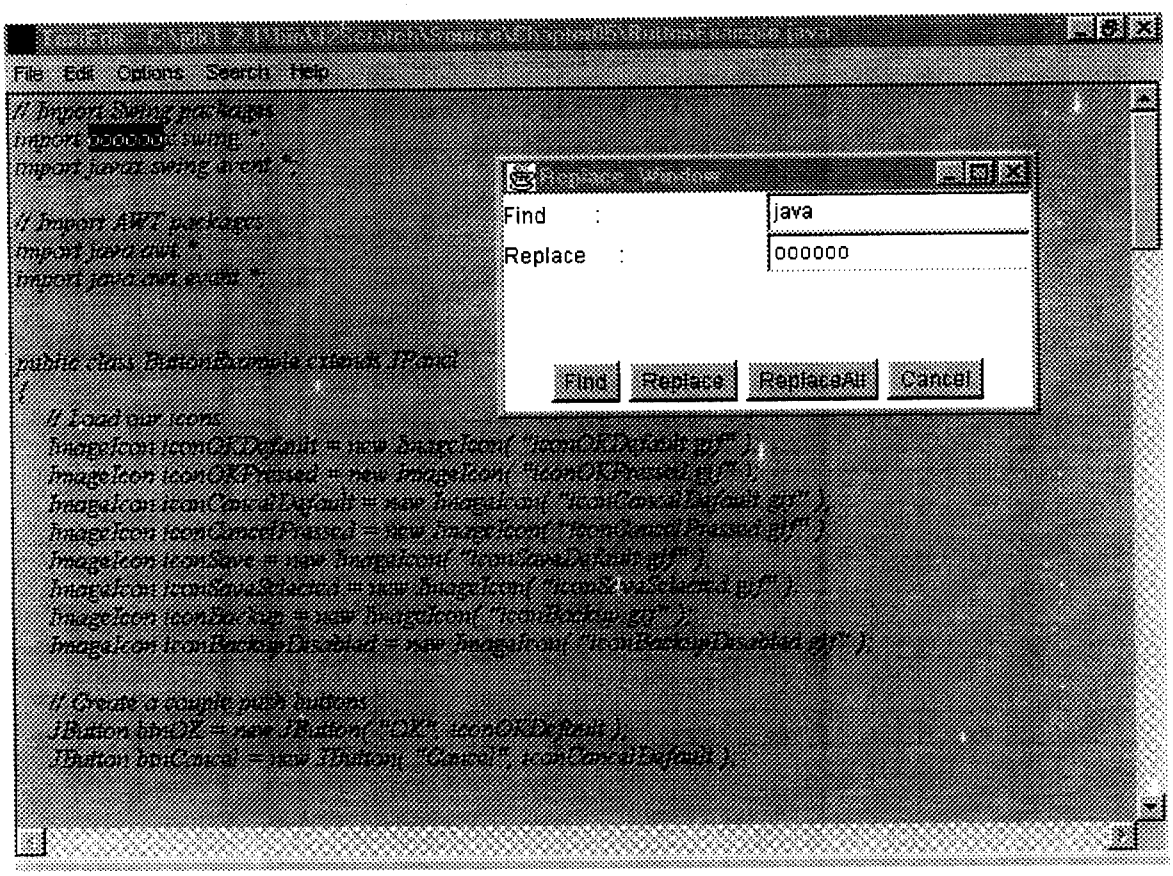
To Move to the specific page



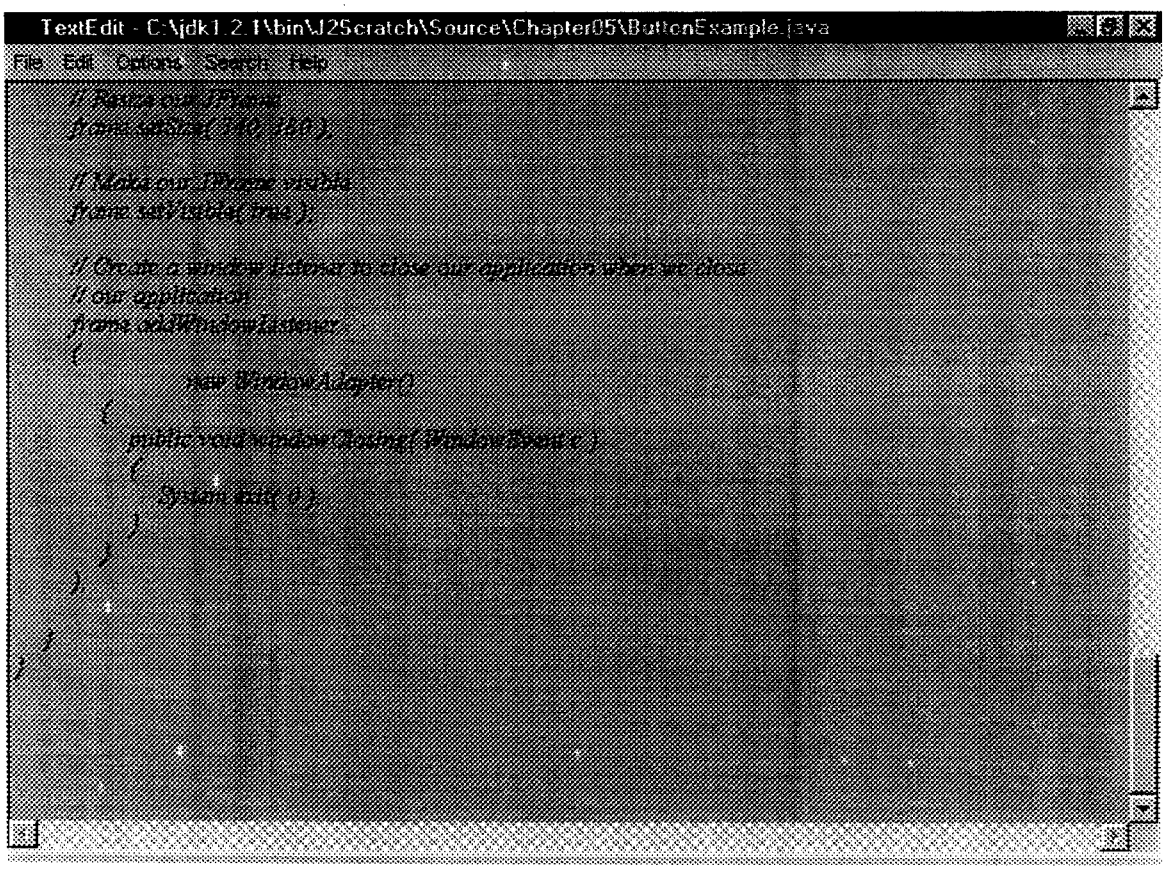
To Search through the keyword



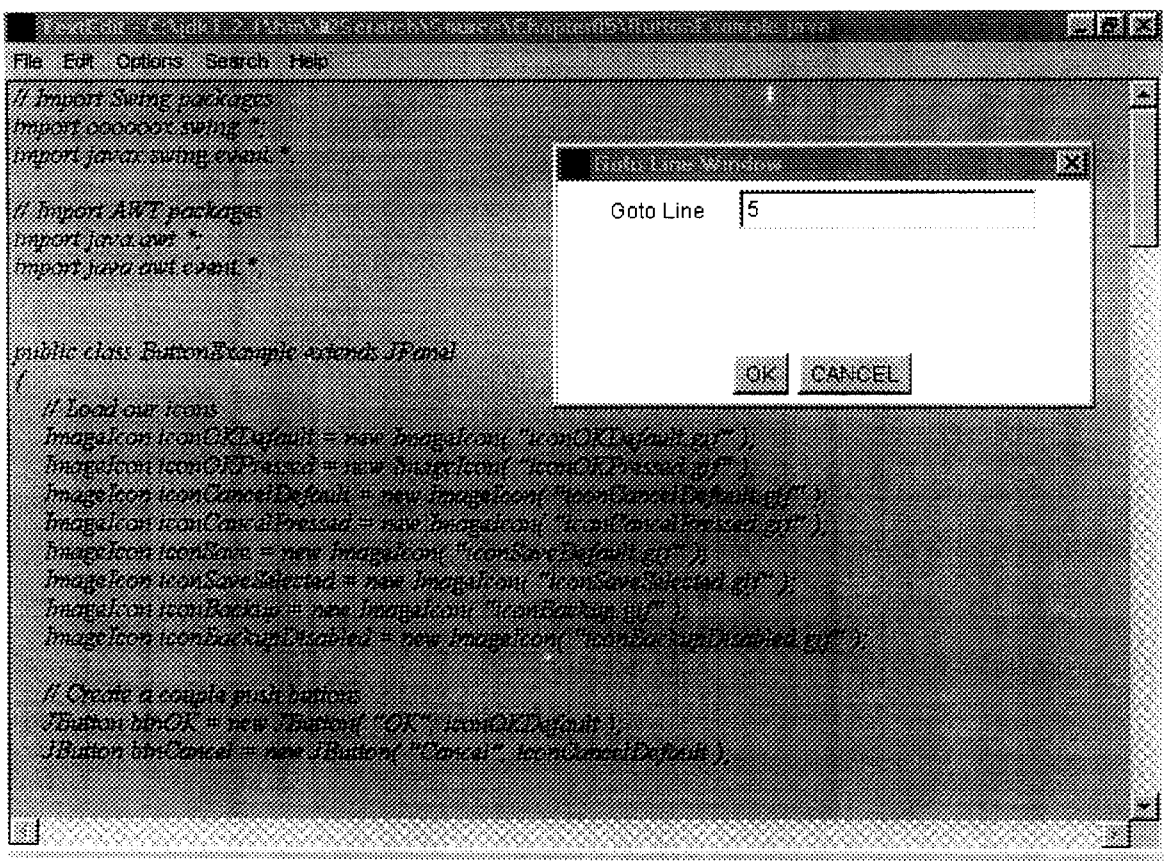
To Replace with the specified keyword



Going to the end of the text

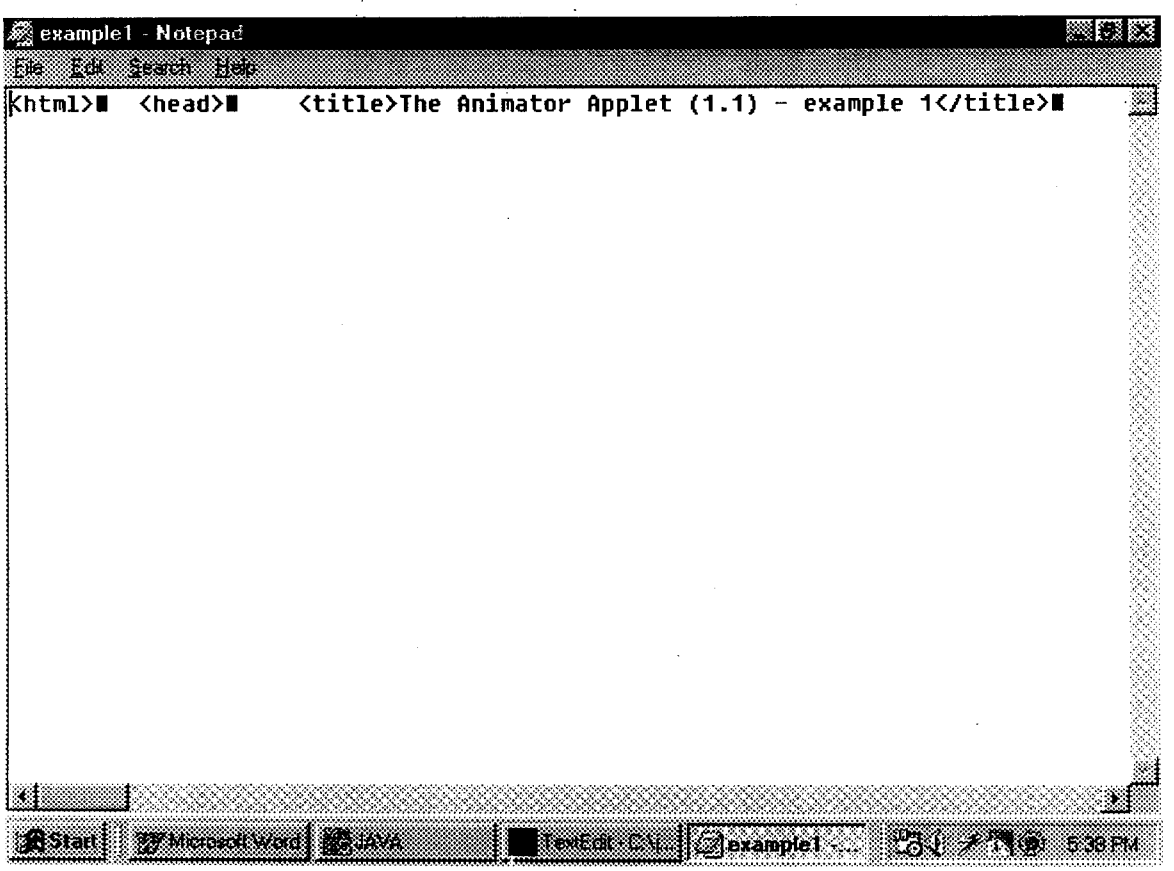


To Goto the specified Line in the Text

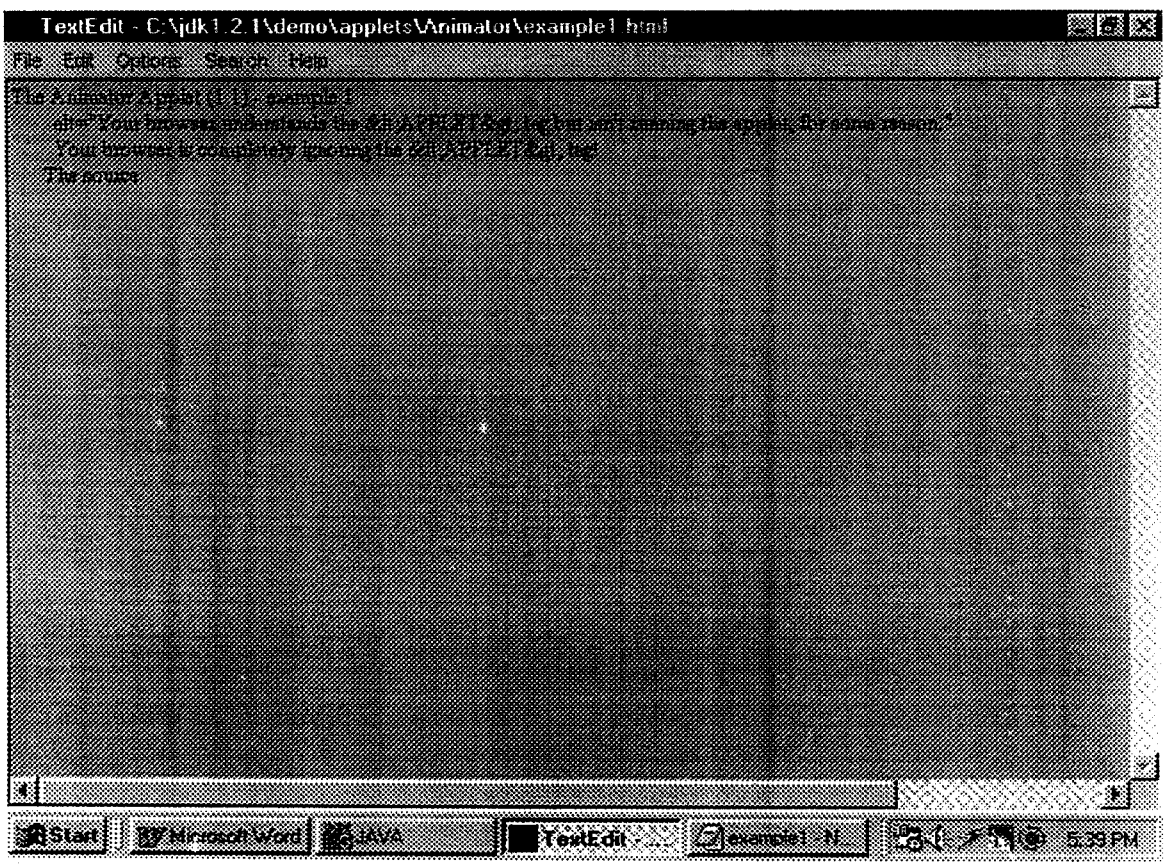


COMPARATIVE FORMATS IN NOTEPAD TO REPORT VIEWER

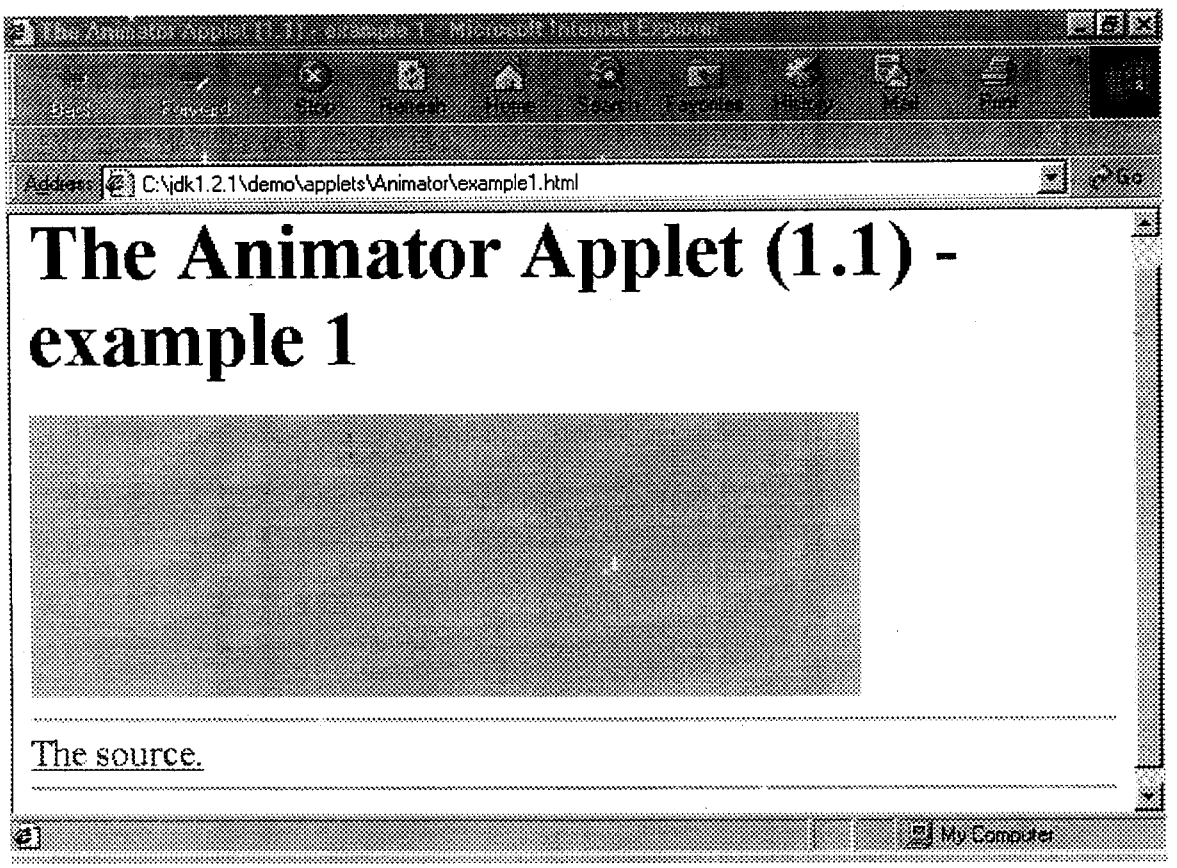
The html file - example1.html in view with Notepad



The HTML file example1.html in my Report Viewer



The same file in the Internet Explorer:



The RTF file Document.rtf in view with wordPad