A

Project Report

on

Text Editor Application

Submitted by:

[DURGESH KUMAR KAUSHIK]

[Roll No : 2120140080058]

Submitted to:

Department of Computer Application

A. N. College, Patna

Patliputra University, Patna

GUIDE CERTIFICATE

This is to certify that the project entitled **"Text Editor Application"** submitted for the partial fulfilment of the requirements for the degree of the **Bachelor of Computer Application** of Patliputra University, Patna.

Mr. Durgesh kumar kaushik worked under my supervision and guidance and that no part of thisreport has been submitted for the award of any other degree, diploma, fellowship or other similar title or prizes and that the work has been published in a journal or magazine.

Acknowledgements

In Successfully Completing this project, many people helped us. We would like to thank all those who are related to this project.

Firstly, we would like to express our special thanks of gratitude to Mr. **Sunil Kumar** as well as our Head of department **Dr. Manish Kumar** who gave us the golden opportunity to do this wonderful project on the topic "**Text Editor Application"**, which also helped us in doing a lot of research and we came to know about so many new things we are really thankful to them. His constant guidance, encouragement and valuable feedback led us to this successful completion of this project.

We learnt a lot about this project. There directions and suggestions helped in the completion of this project.

Finally, we would also like to thank our parents and friends who helped us a lot in finalizing this project within the limited time frame.

Table of Contents

List of Figure

Abstract

There are a lot of text editors available, those that run in the terminal, in a GUI, in a browser, and in a browser engine. Many are very good, and some are great. But sometimes, the most satisfying answer to any question is the one we build ourself.

The Text editor application is Menu driven application which provide popup menu that perform text editing operations, such as file, edit and search operation. A text editor refers to any form of computer program that enables users to create, change, edit, open and view plain text files.

The aim of text editors is to edit text. Computer programs, web pages, web apps, and a great many other things are built out of text. Text editors are also popular for writers who don't want to risk vendor lock-in. Text is open, easily searchable and future proof.

Introduction of Text Editor

✞ Project Description

Building a really good text editor is a lot harder than it may seem. But then again, it’s also not as hard as we might fear to build a basic one. In fact, most programming toolkits already have most of the text editor parts ready for us to use. The components around the text editing, such as a menu bar, file chooser dialogues, and so on, are easy to drop into place. As a result, a basic text editor is a surprisingly fun and elucidating, though intermediate lesson in programming. We might find ourself eager to use a tool of our own construction, and the more we use it, the more we might be inspired to add to it, learning even more about the programming language we’re using.

Text editor is a computer program that lets a user enter, change, store, and usually print text (characters and numbers, each encoded by the computer and its input and output devices, arranged to have meaning to users or to other programs). It allows us to open, view, and edit plain text files.

Typically, a text editor provides an "empty" display screen (or "scrollable page") with a fixed-line length and visible line numbers. We can then fill the lines in with text, line by line. A special command line lets you move to a new page, scroll forward or backward, make global changes in the document, save the document, and perform other

actions. After saving a document, we can then print it or display it. Before printing or displaying it, we may be able to format it for some specific output device or class of output device. Text editors can be used to enter program language source statements or to create documents such as technical manuals.

Software programs and web developers use text editors to write and edit code. This is one of primary purpose pf text editors to help these users to read and write code. If we are a web developer or java developer then text editor is one of best tool for writing our code in efficient way. Text editors generally don't pretend to have any extra knowledge of what we're doing or bells-and-whistles bolted on, in the way an integrated development environment might. An IDE may contain a compiler or a series of language-specific helper functions, buttons and gizmos to do all sorts of things for us. A text editor doesn't usually do that, because it's not required for the core purpose of editing text. As a result, text editors are more lightweight - they start up faster and don't use up so much memory.

Some people use text editors exclusively. If we've got the hang of using Vim for example, we might not want to go back to using anything else, ever.

The aim of text editors is to edit text. Computer programs, web pages, web apps, and a great many other things are built out of text. Text editors are also popular for writers who don’t want to risk vendor lock-in. Text is open, easily searchable, and future proof.

✞ Motivation

A text editor is one of the most popular and relevant pieces of software. And that still applies today, because making software nowadays involves using plenty of tools that increase efficiency and product quality. Test automation software, APIs, and frameworks are among these helpful tools. Each one has its own uses and benefits, and having a combination of that in our development can be beneficial. But today, we’ll be talking about text editors and why they’re still important in 2023. Every text editor has unique characteristics, but all of that have common editing features. These help developers process text and layouts (like in word applications), add/edit text and other files, collaborate, and more. There are some of the reasons why we need a text editor in our application in 2023.

- It improves user experience by going beyond regular text editing.
- It has a place on most websites and web apps.

- It allows non-technical users to build pages or make content with little or no coding.

✞ End Users

Text editor software is used by programmers and developers for manipulating plain text source code. Text editors are ideally fast and lightweight for editing and manipulating a small number of text files at a time, and they provide features like regular expression search. Developers who do not work in a system where a full-fledged IDE is needed, or where an IDE is too restricting, will often find that a text editor gives them more flexibility and freedom to code outside of standardized methods of development.

On the other hand, developers and technical business owners can implement their own text editors in their content-heavy applications for others to use.

Objective

A text editor refers to any form of computer program that enables users to create, change, edit, open and view plain text files. They come already installed on most operating systems but their dominant application has evolved from notetaking and creating documents to crafting complex code. Today, text editors are a core part of a developer's toolbox and are most commonly used to create computer programs, edit hypertext markup language (HTML), and build and design web pages.

✈Main objectives behind creating this text editor are following: -

- A light weight Text Editor Application.
- Easily portable application.
- Fast file loading application.
- Require less system resources.
- Improvement options are also available.
- It enables users to create, change, edit, open and view plain text files.

✈It led us to save our notes exactly (as text files) where we want.

Related Work

✞ Existing System

The existing system for writing notes is like pen-paper or existing editor like word, note++, vscode, etc. In case of paper, it takes time to copy some information on it, it is also hard to manage it like is it at proper place, save it from damage, also search that paper back from stack and also takes time in search particular text from a single or bundle of paper. In case of existing editor, they are huge, complex to use, some have also crashing issue, while some don't support low configuration system and some take a lot of time in loading and saving files.

✞ Proposed System

With the advancement of technology, code editor is really helpful in keeping notes digitally. It makes easy to search through notes and easily available. It takes less time in loading due to less codes and also saves file faster. It provides search feature which helps in finding particular

text from the file. It provides different editing feature to view a particular file. Overall, it helps and ease the text writing and editing.

Requirement Analysis

✞ Requirement Gathering

In the software development life cycle (SDLC),

Requirement analysis is the first step of major importance. The entire concentration is on gathering the functional and the nonfunctional requirements for the product to be developed and estimating the feasibility of those attributes. Through requirement gathering we ensure that we are setting project goals and objectives much earlier.

Complete understanding of the requirements leads to the successful development of the software. If we don't do this step, then however hard we work we will never arrive at the desired final product.

This is most crucial as without knowing the exact requirements the final output can never be achieved as desired. For this project, I did major research on the existing system and discussed the functionality that I wanted to develop with my major professor and finally

concluded on a concrete set of requirements that I wanted to see as an outcome of my project.

✞ Functional Requirements

The proposed system has the following functionalities:

- User Can Create new file
- User Can open existing file
- User Can save the file in desired folder
- User Can exit from text editor
- User Can cut, copy and paste the text in text editor application.
- User Can undo the text.
- User Can select the all text and perform the required text formatting.
- User Can delete the text in text editor application.
- User have facility to change the size of text and set desired font.
- User Can search specific text and move on next text.
- User have facility to know about the text editor application.

✞ Non-Functional Requirements

The major difference between Functional and Non-functional difference lies in two keywords. “What” and “How”. While what the system does is described by its

functional requirements, how the system achieves those quality attributes are discussed by the nonfunctional requirements. The non-functional requirements focus on the scalability of the system, security of the system, reliability and maintenance of the system which are ensured by verifying and validating the system. The proposed system meets the non-functional requirements like edited text is valid, editor is reliable, controlling access of the users. The application is made more scalable with an advanced search feature that delivers information.

✞ Requirement Specifications

Requirement Specification is an important step in the software development life cycle. Although various software and hardware specifications can be used to develop this application, the software and hardware specifications that I have used to develop this job search portal are mentioned below:

✪ Software Requirements

Operating System: Windows10 & Higher

Compiled Used : Java version 7

Front End : Java swing, AWT

○ Hardware Requirements

Architecture : 32 Bit(x86)/64Bit(x64)

Memory (RAM) : 512MB of RAM required.

Hard Disk Space : 100 MB of Storage Space required.

Processor : Intel core i5 and newer, Intel Pentium4, AMD.

Basic Themes of Project

The Text editor application is Menu driven application which provide popup menu that perform text editing operations, such as file, edit and search operation. A text editor refers to any form of computer program that enables users to create, change, edit, open and view plain text files.

We are going to give menu driven application that have following operation will be considered.

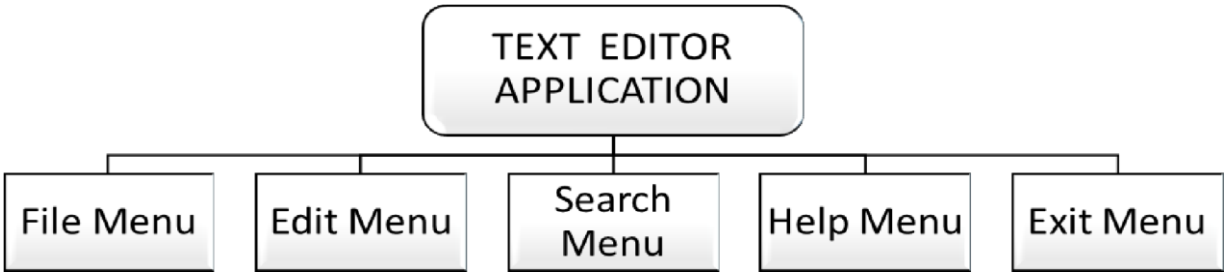
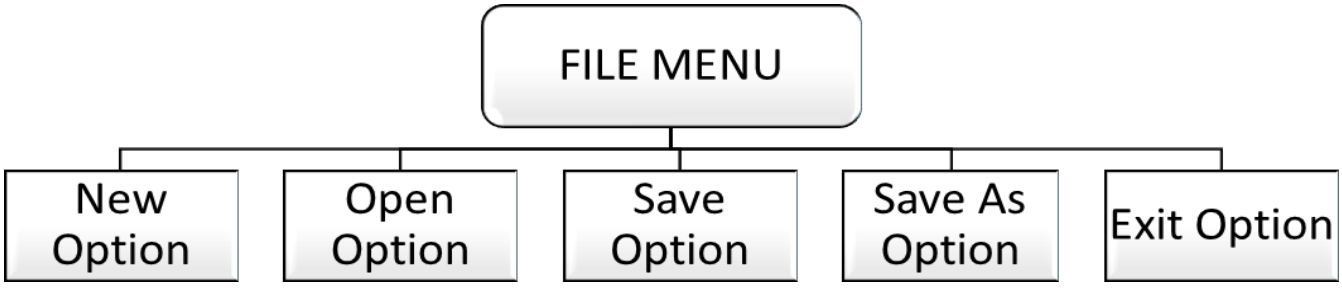


fig1 - Text File Application

✞ **File Menu: -** The file operations include creating a new file, opening an existing file, saving a file and exiting the application.



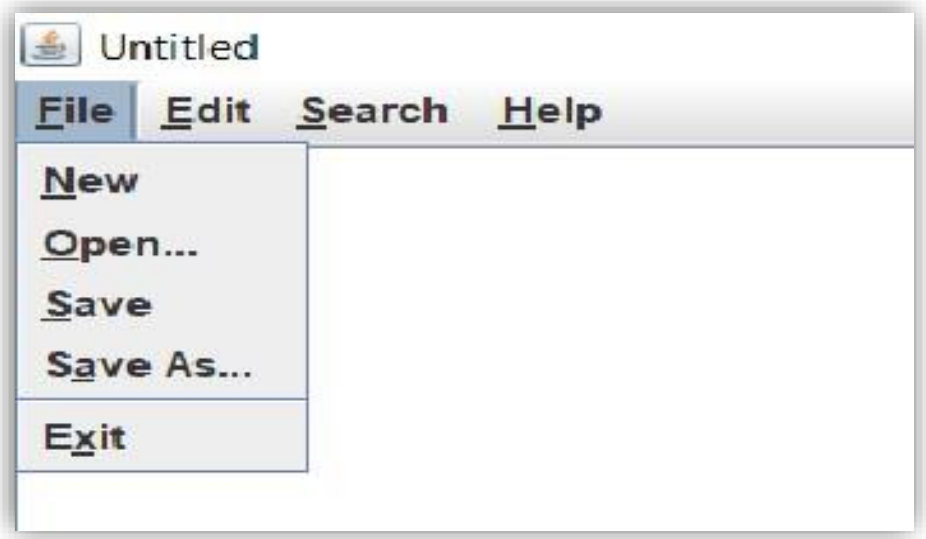
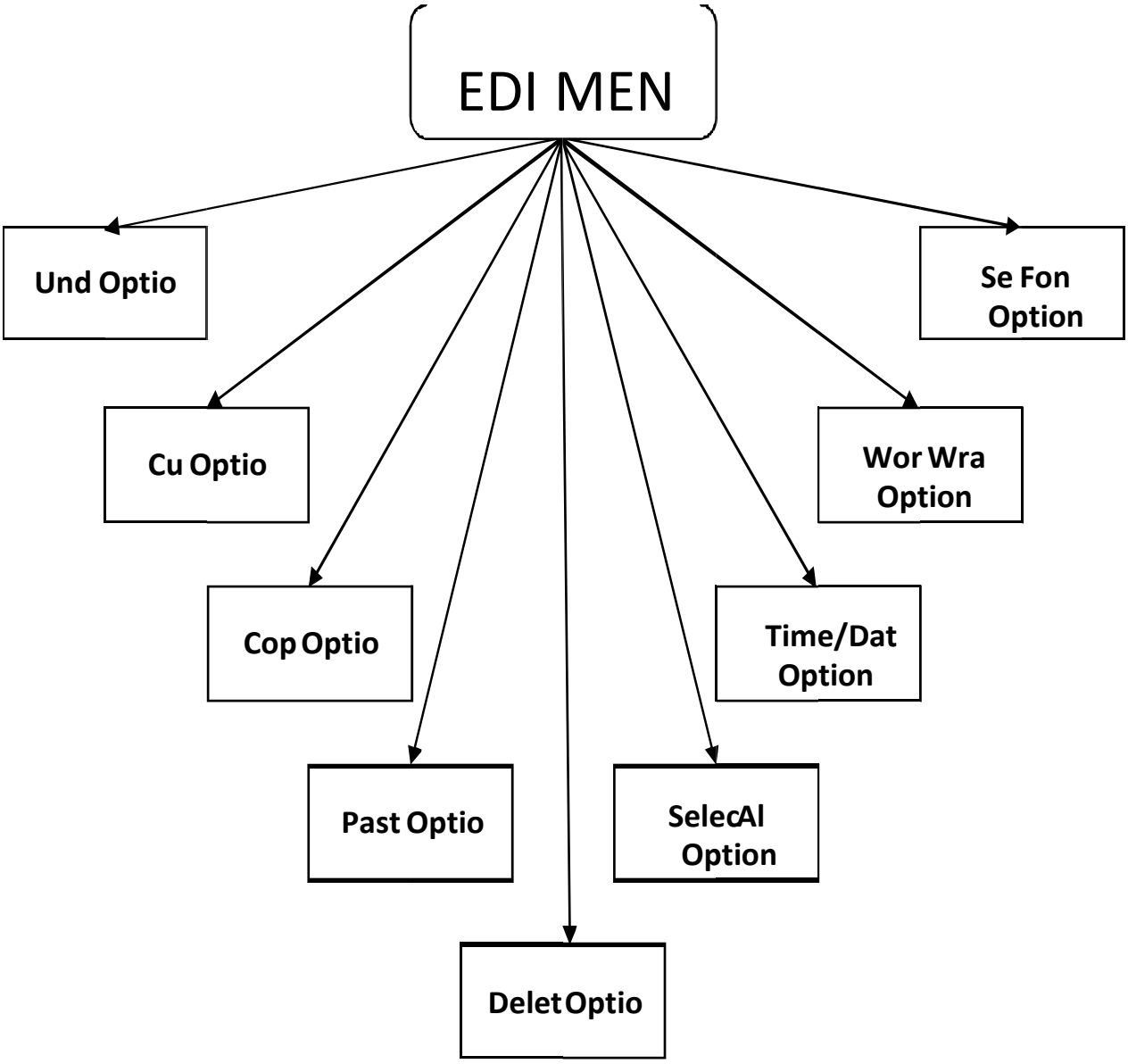


fig2 - File menu

✞ **Editing Operations:** - The edit operation includes undo, cut, copy, paste, delete, select All, time/date, word wrap and set font oration.


EDI MEN
Und Optio
Se Fon Option
Cu Optio
Wor Wra Option
Cop Optio
Time/Dat Option
Past Optio
SelecAl Option
DeletOptio

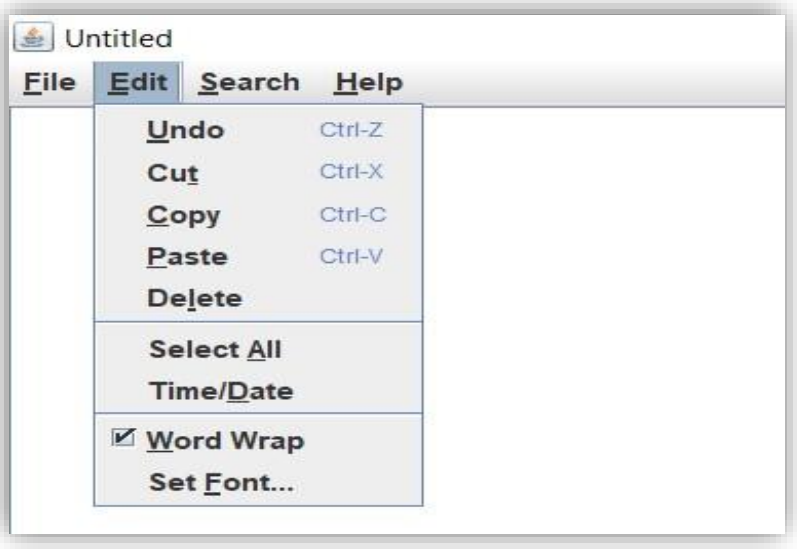


fig3 - Edit menu

✞ **Help Menu:** - It is a part of our text editor application that allows the user to access information about how to use this application.

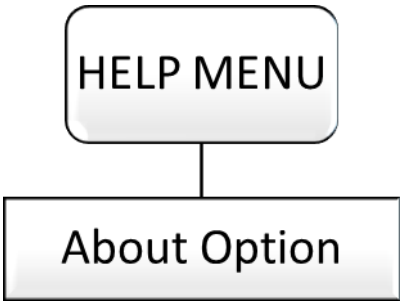


fig4 - Help menu

✞ **Search Operation:** - The search operation includes find and find next operations. This enables us to search for text strings in files. We can also search for objects within a configuration using a text search.

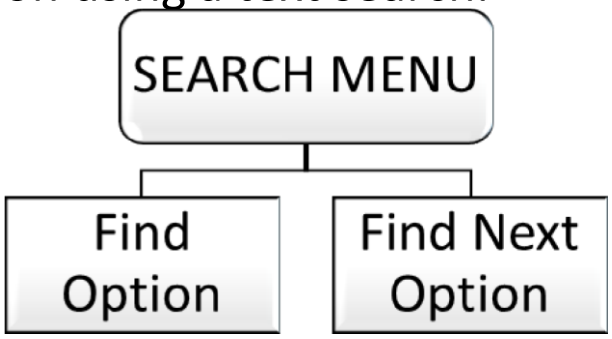


fig5 - Search menu

Menu	Description
File Menu	Includes new, open, save, save as and exit options
Edit Menu	Include undo, cut, copy, paste, delete, select all, time/date, word wrap, and set font option.
Search Menu	Include find and find next option
Help Menu	Include help options

Technology Used

✞ Java

Java is a multi-platformed, object-oriented, and network-centric language that can be used as a platform in itself. It is a fast, secure, reliable programming language for coding everything from applications (like text editor), mobile apps, and server-side technologies. Java is a widely-used programming language for coding web applications. It has been a popular choice among developers.

✞ OOPS Concept

Object-Oriented Programming (OOP) is a programming model that use classes and objects. It's utilized to break down a software program into reusable code blueprints (called classes) that we may use to build specific instances of things. Object-oriented programming languages include many programming languages, java is one of them.

Here are some features of OOPS

✞ Objects

An object is a self-contained segment with the attributes and processes needed to make data usable in programming terms. From an object-oriented perspective, objects are the main building pieces of programs.

In each application we create, we may employ a variety of objects of various sorts. Each kind of object is derived from a specific class of that type. Consider an object to be a sculpt of the real-world perceptions, processes, or objects that are important to the application we're designing.

✞ Classes

In the oops concept, a class is a construct that is used to describe an individual type. The class is instantiated into instances of itself referred to as class instances or simply objects. A class defines ingredient members that allow its instances to have position and behavior. Member variables or instance variables facilitate a class instance to maintain its position.

On the other hand, other kinds of members, especially methods, allow the behavior of class instances. Simply

classes consequently define the type of their instances. A class usually represents a person, place or thing, or something.

✞ Inheritance

The attributes that we inherit from our parents are a simple illustration of inheritance. Classes may inherit characteristics from other classes thanks to inheritance. Parent classes, in other words, extend properties and behaviors to child classes.

Reusability is aided via inheritance. Prototyping is another name for inheritance in JavaScript. A prototype object serves as a base from which another object may derive its features and actions. Thus, you may use multiple prototype object templates to form a prototype chain. Inheritance is passed down from one generation to the next Parent.

✞ Polymorphism

If one task is performed in different ways, it is known as polymorphism. In Java, we use method overloading and method overriding to achieve polymorphism.

✞ Abstraction

Hiding internal details and showing functionality is known as abstraction. In Java, we use abstract class and interface to achieve abstraction.

✞ Encapsulation

Binding (or wrapping) code and data together into a single unit are known as encapsulation. A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

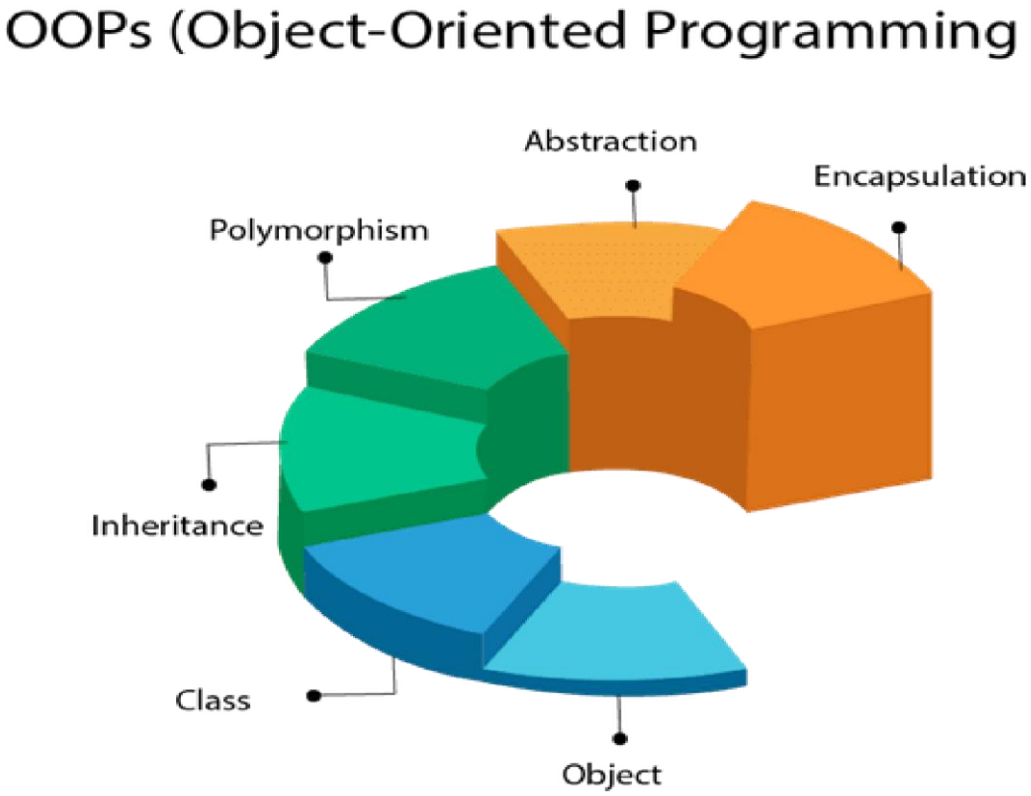


fig6 - OOPs System

✞ Java API

Java defines the syntax and semantics of the Java programming language. This includes the basic vocabulary and rules used to write algorithms such as primitive data types, if/else blocks, loops, etc.

APIs are important software components bundled with the Java Platform. These are pre-written Java programs that can plug and play existing functionality into your own code. For example, you could use Java APIs to get the date and time, perform mathematical operations, or manipulate text.
Any Java application code written by a developer will typically combine new and pre-existing code from Java APIs and Java libraries

✞ Java Swing

Swing is a Java Foundation Classes [JFC] library and an extension of the Abstract Window Toolkit [AWT]. Swing offers much-improved functionality over AWT, new components, expanded components features, excellent event handling with drag and drop support.

Some Applications of Swing Are

- Swing is a Set of API (API- Set of Classes and Interfaces)
- Swing is Provided to Design a Graphical User Interfaces
- Swing is an Extension library to the AWT (Abstract Window Toolkit)
- Includes New and improved Components that have been enhancing the looks and Functionality of GUI's
- It Employs model/view design architecture

Swing Classes Hierarchy

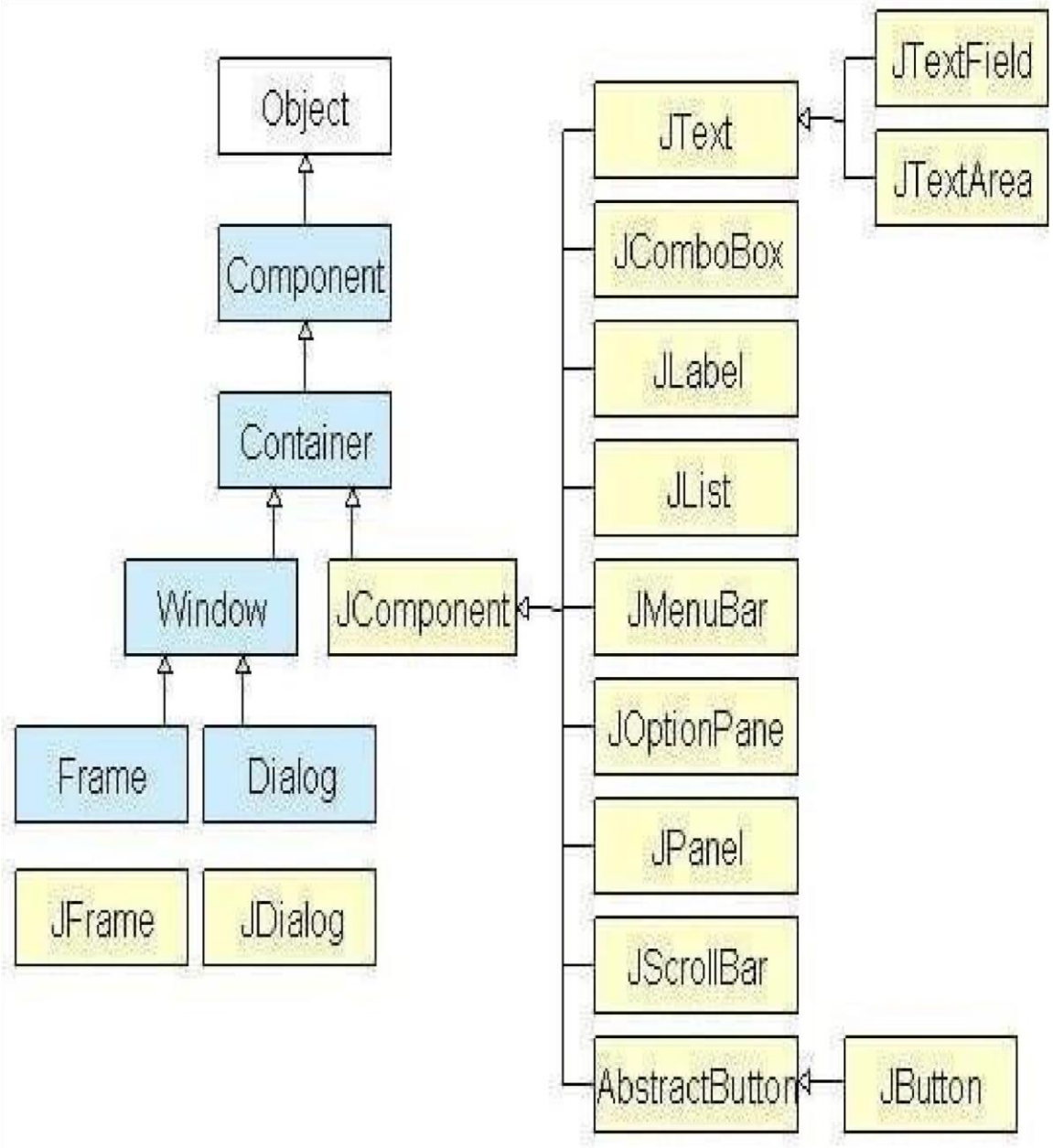


fig7 - Swing Class Hierarchy

✞ JLabel

JLabel is a component which displays a readable text or an image in the Swing Container User interface. The

application user cannot edit the text rendered in the JLabel.

However, the application itself, through action events, can change the text. JLabel component can display both plain and HTML text.

✞ JTextField

JTextField is a swing component that allows users to input one line of text. JTextField inherits from the JTextComponent class of javax.swing Library.

✞ JButton

JButton is one of the swing components which gives swing the property of platform independence. This component creates a click effect on the application's user interface. It is implemented in an application by calling any of its class constructors.

✞

JTextArea

JTextArea class renders a multi-line text box. Similar to the JTextField, a user can input non-formatted text in the field. The constructor for JTextArea also expects two integer parameters which define the height and width of the text-area in columns. It does not restrict the number of characters that the user can input in the text-area.

✞ JMenuBar, JMenu and JMenuItem

In Java, Swing toolkit contains a JMenuBar, JMenu and JMenuItem class. It is under package javax.swing.JMenuBar, javax.swing.JMenu and javax.swing.JMenuItem class. The JMenuBar class is used for displaying menubar on the frame. The JMenu Object is used for pulling down the components of the menu bar. The JMenuItem Object is used for adding the labelled menu item.

✞ AWT Package

AWT stands for Abstract window toolkit is an application programming interface (API) for creating Graphical User Interface (GUI) in Java. It allows Java programmers to

develop window-based applications. AWT provides various components like button, label, checkbox, etc. used as objects inside a Java Program.

✞ AWT features include

- A set of native user interface components.
- A robust event-handling model
- Graphics and imaging tools, including shape, color, and font classes.
- Layout managers, for flexible window layouts that do not depend on a particular window size or screen resolution.
- Data transfer classes, for cut-and-paste through the native platform clipboard.

Some AWT Classes used by this project are

✞ java.awt.Frame class

A Frame is a top-level window with a title and a border. The default layout for a frame is BorderLayout. Frames are capable of generating the following types of window

events: WindowOpened, WindowClosing, WindowClosed, WindowIconified, WindowDeiconified, WindowActivated, WindowDeactivated.

setTitle()

This is the method of java.awt.Frame class, which is used to set the title of frame. We can change the variable title but it doesn't affect the frame.

We will need to call setTitle on the frame again. In the constructor, assign the new JFrame to the instance variable, so we can change its title later in setTitle() method.

Set the title of the frame in the setTitle () method:

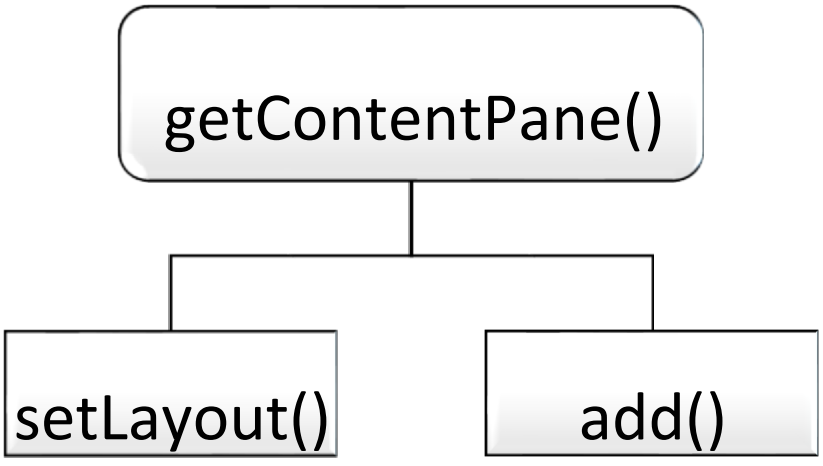
```
JFrame frame1 = new JFrame;
public void setTitle(String name)
{
   frame1.setTitle("Java");
}
```

✞

getContentPane ()

The getContentPane() method retrieves the content pane layer so that we can add an object to it. The content pane is an object created by the Java run time environment. A container has several layers in it. You can think of a layer as a transparent film that overlays the container. The layer that is used to hold objects is called the content pane. Objects are added to the content pane layer of the container.

The getContentPane () method retrieves the content pane layer so that we can add an object to it.getContentPane() returns a container to hold objects. You can add objects on the returned container instead of adding objects directly to the JFrame or JDialog.



setLayout()

✞

Every Abstract Window Toolkit (AWT) and Swing container has a predefined layout manager as its default. It is easy to use the container. setLayout method to change the layout manager, and we can define our own layout manager by implementing the java.awt.

The setLayout() method allows us to set the layout of the container, often a JPanel, to say FlowLayout,

BorderLayout, GridLayout, null layout, or whatever layout desired. The layout manager helps layout the components held by this container.

✞ **add ()**

In order to adding a component to a container, firstly we create an instance of the desired component and then call the add () method of the Container class to add it to a window. The add () method has many forms and one of these is.

Component add (Component c)

This method adds an instance of component (i.e., c) to the container. The component added is automatically visible whenever its parent window is displayed.

Event Handling in AWT

In general, we cannot perform any operation on dummy GUI screen even any button clicks or select any item. To perform some operation on these dummy GUI screen we need some predefined classes and interfaces. All these types of classes and interfaces are available in java.awt.event package.

Changing the state of an object is known as an event. The process of handling the request in GUI screen is known as event handling (event represent an action). It will be changes component to component.

In event handling mechanism event represent an action class and Listener represent an interface. Listener interface always contains abstract methods so here you need to write your own logic.

AWT Event Listener Interfaces

The Event listener represent the interfaces responsible to handle events. Java provides us various Event listener classes but we will discuss those which are more frequently used in our project. Every method of an event

✞

listener method has a single argument as an object which is subclass of EventObject class.

For example, mouse event listener methods will accept instance of MouseEvent, where MouseEvent derives from EventObject.

It is a marker interface which every listener interface has to extend. This class is defined in java.util package.

AWT ActionListener Interface

The class which processes the ActionEvent should implement this interface. The object of that class must be registered with a component. The object can be registered using the addActionListener () method. When the action

✞

event occurs, that object's actionPerformed method is invoked.

✞Interface methods

void **actionPerformed** (ActionEvent e)

✞Methods inherited

Java.awt.EventListener

✞ Interface declaration

public interface ActionListener
extends EventListener

AWT ComponentListener Interface

The class which processes the ComponentEvent should implement this interface. The object of that class must be registered with a component. The object can be registered using the addComponentLis tener () method. Component event are raised for information only.

✞ Interface declaration

Methods inherited

. **java.awt.EventListener**

✞ Interface methods

```
public interface ComponentListener
  extends EventListener
```

✞

S.N.	Method & Description
1	Void componentHidden (ComponentEvent e)
2	void componentMoved (ComponentEvent e)
3	void componentResized (ComponentEvent e)
4	void componentShown (ComponentEvent e)

Java IO: Input-output in Java

Java brings various Streams with its I/O package that helps the us to perform all the input-output operations. These streams support all the types of objects, data-types, characters, files etc to fully execute the I/O operations.

It is used *to process the input* and *produce the output*.

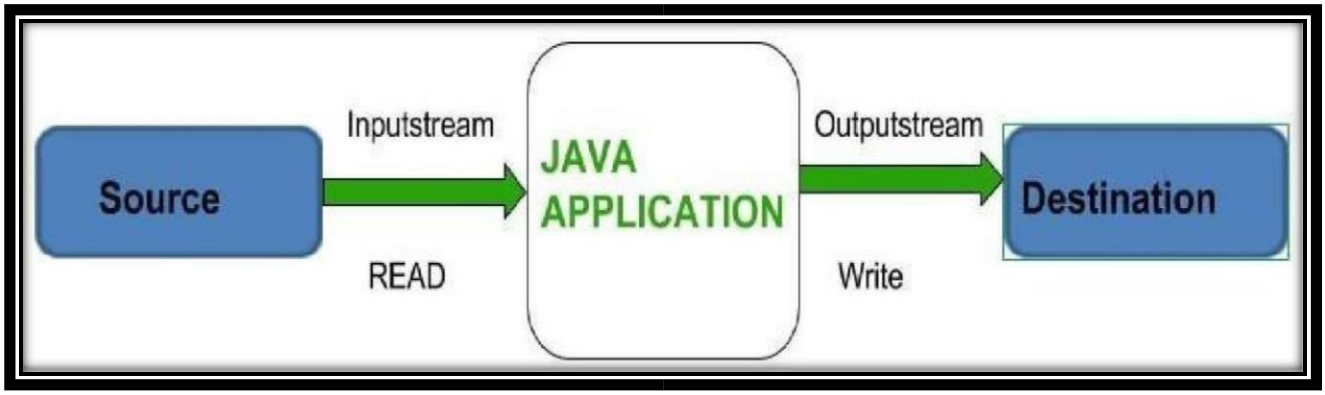


Fig8 - Input/Output in Java

✞

Java I/O package contain various classes which is used by programmer to developed their project or other purpose. In this project we also use some classes that is listed below.

FileReader Class

Java FileReader class is used to read data from the file. It returns data in byte format like FileInputStream class.

It is character-oriented class which is used for file handling in java.

✞ FileReader class declaration

```
public class FileReader extends InputStreamReader
```

✞ FileWriter Class

Java FileWriter class is used to write character-oriented data to a file. It is character-oriented class which is used for file handling in java.

✞

Unlike FileOutputStream class, we don't need to convert string into byte array because it provides method to write string directly.

FileWriter class declaration

```
public class FileWriter extends OutputStreamWriter
```

✞ BufferedReader Class

Java BufferedReader class is used to read the text from a character-based input stream. It can be used to read data line by line by readLine () method. It makes the performance fast. It inherits Reader class.

✞ BufferedReader class declaration

```
public class BufferedReader extends Reader
```

✞ Util Package

✞

Java.util package contains the collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes. This reference will take us through simple and practical methods available in java.util package.

java.util.Date Class

The java.util.Date class represents date and time in java. It provides constructors and methods to deal with date and time in java.

The java.util.Date class implements Serializable, Cloneable and Comparable<Date> interface.

✞ Class declaration

```
public class Date
   extends Object
   implements Serializable, Cloneable,
Comparable<Date>
```

Class constructors

Sr. No.	**Constructor & Description**
	Date ()

✞	
This constructor allocates a Date object 1. and initializes it so that it represents the time at which it was allocated, measured to the nearest millisecond.	
Date (long date) This constructor allocates a Date object and initializes it to represent the 2. specified number of milliseconds since the standard base time known as "the epoch", namely January 1, 1970, 00:00:00 GMT.	

✞ Why we use java for this project?

We wanted to develop a GUI application for PC's specially for Windows. We had many options for programming language like C++, Vb.net etc. but we selected java because java is a complete object-oriented programming language with too many free supports and facilities. Java is completely handy with GUI application development with its libraries like "AWT, Swing. Also, java have

Complete library of event handling which help a lot while we patching different events with different parts of the program.

There are mainly three libraries used for creating GUI through java in which we used AWT and swing for our project because they are the oldest one and have a lot of supports for completing our project. We also used some other libraries in our project to fulfill the requirements of the project. They are IO, Util, Lang etc. After completing this project, we found that selecting java as project language was a write decision and we learnt a lot more about java and its features.

Use Case Diagram

While understanding only the static nature of a system is insufficient, Use-Case diagrams helps to give the dynamic view of the system. Use Case diagrams models the system and the subsystems of an application.

There are some external and internal factors that marks the dynamic nature of the Use Case diagram. We call them actors. While Use case diagrams can be considered as a high-level requirement analysis of the system, they give a clear notion of the actors and their roles (use cases) and hence is an important pictorial representation to understand system specifications early in the project. Use case diagrams are a clear visualization of actors (the internal or external factors), their roles (use cases) and relationship amongst these actors and their roles.

Data Flow Diagram (DFD)

A data flow diagram (DFD) is a graphical or visual representation using a standardized set of symbols and notations to describe a business's operations through data movement.

The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have

control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart. Data Flow Diagram can be represented in several ways. The DFD belongs to structured-analysis modeling tools. Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in software-system processes.

✞ Levels of DFD

A data flow diagram can dive into progressively more detail by using levels, zeroing in on a particular piece. DFD levels are numbered 0, 1 or 2, and occasionally go to even Level 3 or beyond.

DFD uses hierarchy to maintain transparency thus multilevel DFD's can be created.

Levels of DFD are as follows:

✞ 0-level DFD

It is also known as a context diagram. It's designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.

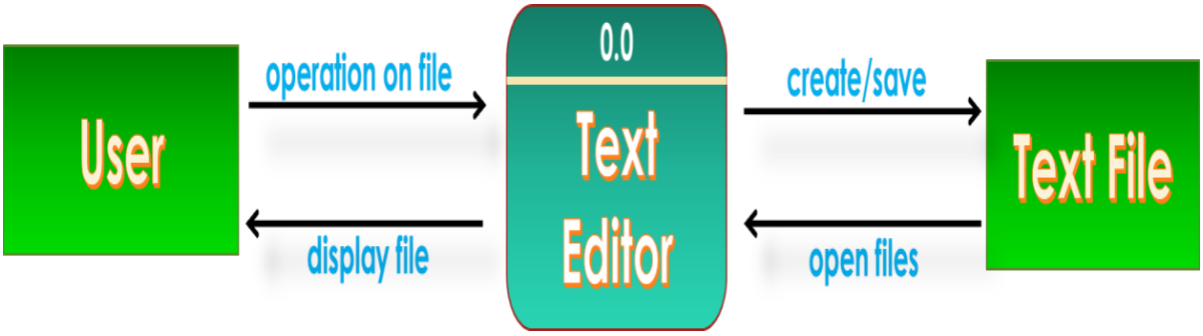


fig9 - 0 Level Diagram

✞ 1-level DFD

In 1-level DFD, the context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main functions of the system and breakdown the high-level process of 0-level DFD into subprocesses.

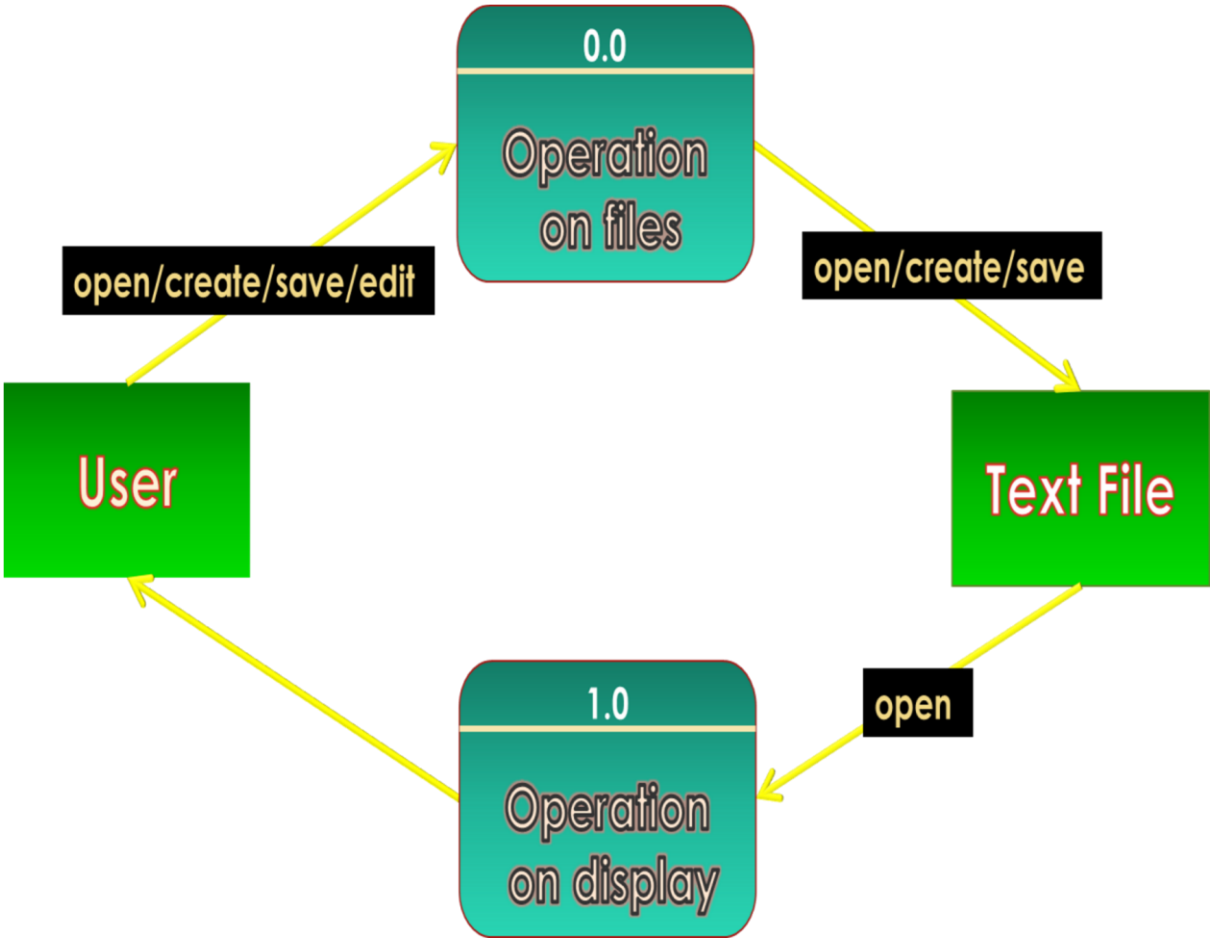


fig10 - 1 Level Diagram

✞ 2-level DFD

2- level DFD goes one step deeper into parts of 1-level DFD. It can be used to plan or record the specific/necessary detail about the system's functioning.

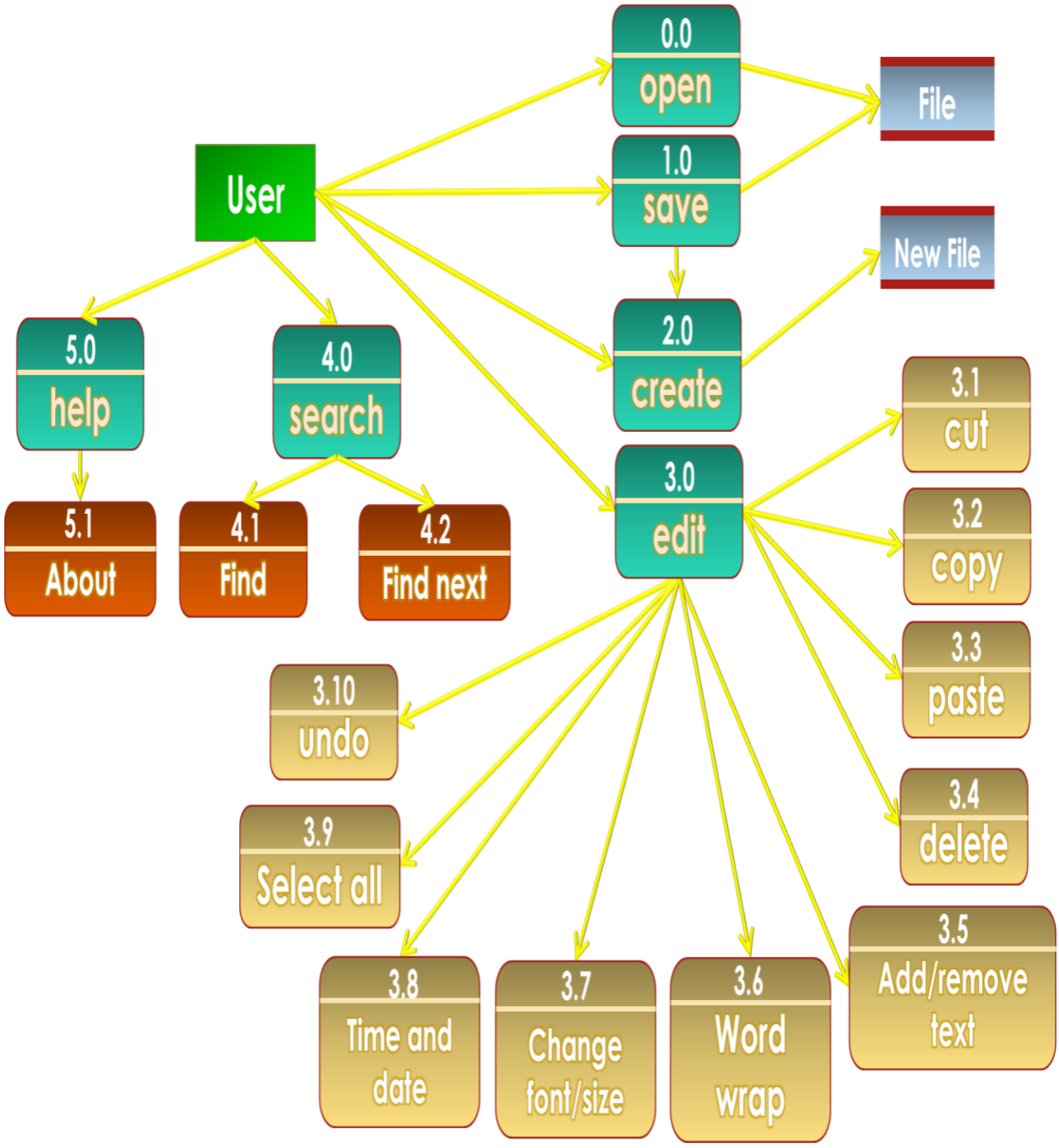


fig11 - 2 Level Diagram

Entity Relationship Diagram

An Entity Relationship (ER) Diagram is a type of flowchart or graphical representation of data that illustrates how "entities" such as people, objects or concepts relate to

each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research.

It also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes.

Following are the main components and its symbols in ER Diagrams:

- **Rectangles:** This Entity Relationship Diagram symbol represents entity types.
- **Ellipses:** Symbol represent attributes.
- **Diamonds:** This symbol represents relationship types.
- **Lines:** It links attributes to entity types and entity types with other relationship types.
- **Primary key:** Attributes are underlined.

They mirror grammatical structure, with entities as nouns and relationships as verbs.
An ERD contains different symbols and connectors that visualize two important information.

1) The major entities within the system scope.
2) Inter-relationships among these entities.

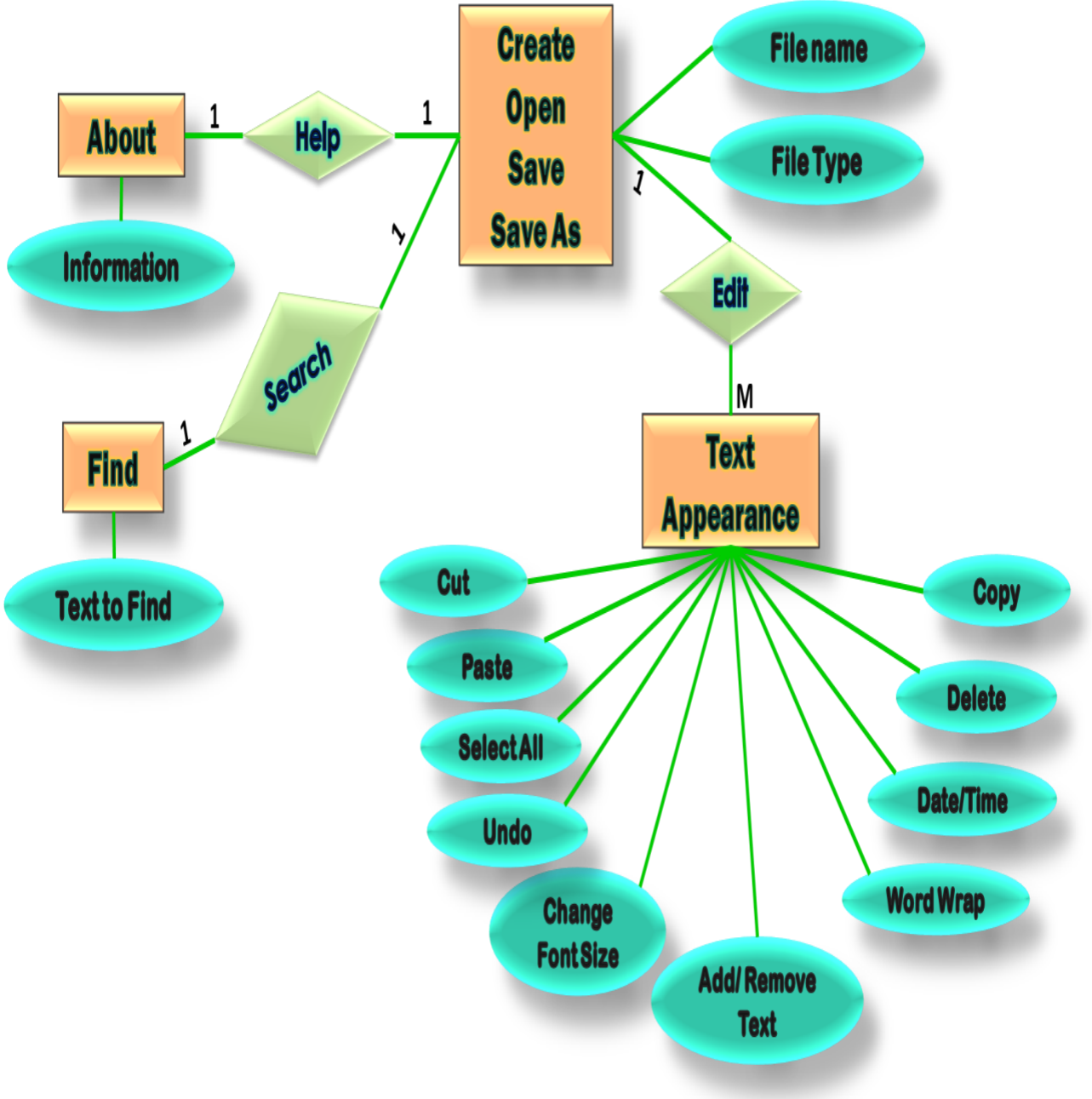


fig12 - ER Diagram

IMP□RTED

```
//THE IMPORTED LIBRARIES USED IN THIS CODE ARE
FOLLOWING
import javax.swing.*;
import javax.swing.undo.*;
import javax.swing.event.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.io.*;
```

MAIN

```
// MAIN CLASS OF THE PROGRAM
public class TextEditer extends JFrame implements
ActionListener {
```

DECLARATION OF OBJECTS & VARIABLES

INSIDE MAIN CLASS

```
// STATIC REFERENCE OF MAIN CLASS

public static TextEditer e;

// DECLARATION OF ALL THE VARIABLES USED IN THIS
// APPLICATION
JTextArea text = new JTextArea(0, 0);
JScrollPane scroll = new JScrollPane(text);

JMenuBar mb = new JMenuBar();

JMenu FILE = new JMenu("File");
JMenu EDIT = new JMenu("Edit");
JMenu SEARCH = new JMenu("Search");
JMenu HELP = new JMenu("Help");

JMenuItem NEWFILE = new JMenuItem("New");
JMenuItem OPENFILE = new JMenuItem("Open...");
JMenuItem SAVEFILE = new JMenuItem("Save");
JMenuItem SAVEASFILE = new JMenuItem("Save As...");
JMenuItem EXITFILE = new JMenuItem("Exit");
```

```
JMenuItem UNDOEDIT = new JMenuItem("Undo");
JMenuItem CUTEDIT = new JMenuItem("Cut");
JMenuItem COPYEDIT = new JMenuItem("Copy");
JMenuItem PASTEDIT = new JMenuItem("Paste");
JMenuItem DELETEDIT = new JMenuItem("Delete");
JMenuItem SELECTEDIT = new JMenuItem("Select All");
JMenuItem TIMEDIT = new JMenuItem("Time/Date");
JCheckBoxMenuItem WORDEDIT = new
JCheckBoxMenuItem("Word Wrap");
JMenuItem FONTEDIT = new JMenuItem("Set Font...");

JMenuItem FINDSEARCH = new JMenuItem("Find");
JMenuItem FINDNEXTSEARCH = new JMenuItem("Find Next");

JMenuItem ABOUTHELP = new JMenuItem("About");

JPopupMenu POPUP = new JPopupMenu();
JMenuItem UNDOPOPUP = new JMenuItem("Undo");
JMenuItem CUTPOPUP = new JMenuItem("Cut");
JMenuItem COPYPOPUP = new JMenuItem("Copy");
JMenuItem PASTEPOPUP = new JMenuItem("Paste");
JMenuItem DELETEPOPUP = new JMenuItem("Delete");
JMenuItem SELECTPOPUP = new JMenuItem("Select All");
```

```
// USER DEFINED CLASSES OBJECT.
UndoManager undo = new UndoManager();
UndoAction undoAction = new UndoAction();


boolean opened = fase;
String wholeText, findString, filename = null;
int ind = 0;
```

END OF DECLARATION OF OBJECTS & VARIABLES

CONSTRUCTOR OF MAIN CLASS | **INSIDE MAIN CLASS**

```
// DEFAULT CONSTRUCTOR OF THE TextEditer CLASS
public TextEditer() {
    // SETING DEFAULT TITLE OF THE FRAME
    setTitle("Untitled");

    // SETTING DEFAULT SIZE OF THE FRAME
    setSize(600, 400);

    // MAKING THE FRAME VISIBLE
    setVisible(true);

    // SETTING WORD WRAP TO TRUE AS DEFAULT
    text.setLineWrap(true);

    // SETTING THE DEFAULT STATE OF WORDWRAP MENU
    OPTION IN EDIT MENU
    WORDEDIT.setState(true);

    // SETTING THE LAYOUT OF THE FRAME
    getContentPane().setLayout(new BorderLayout());

    // ADDS THE SCROLLPANE CONTAINING THE TEXTAREA TO
    THE CONTAINER
    getContentPane().add(scroll,
    BorderLayout.CENTER);

    // ADDING THE MAIN MENUBAR TO THE FRAME
    setJMenuBar(mb);
```

```
// ADDING MENUS TO THE MAIN MENUBAR
mb.add(FILE);
mb.add(EDIT);
mb.add(SEARCH);
mb.add(HELP);

// ADDING MENUITEMS TO THE FILE MENU
FILE.add(NEWFILE);
FILE.add(OPENFILE);
FILE.add(SAVEFILE);
FILE.add(SAVEASFILE);
FILE.addSeparator();
FILE.add(EXITFILE);

// ADDING MENUITEMS TO THE EDIT MENU
EDIT.add(UNDOEDIT);
EDIT.add(CUTEDIT);
EDIT.add(COPYEDIT);
EDIT.add(PASTEDIT);
EDIT.add(DELETEDIT);
EDIT.addSeparator();
EDIT.add(SELECTEDIT);
EDIT.add(TIMEDIT);
EDIT.addSeparator();
EDIT.add(WORDEDIT);
EDIT.add(FONTEDIT);
```

```
// ADDING MENUITEMS TO THE SEARCH MENU
SEARCH.add(FINDSEARCH);
SEARCH.add(FINDNEXTSEARCH);

// ADDING MENUITEM TO THE HELP MENU
HELP.add(ABOUTHELP);

// ADDING MENUITEMS TO THE POPUPMENU
POPUP.add(UNDOPOPUP);
POPUP.addSeparator();
POPUP.add(CUTPOPUP);
POPUP.add(COPYPOPUP);
POPUP.add(PASTEPOPUP);
POPUP.add(DELETEPOPUP);
POPUP.addSeparator();
POPUP.add(SELECTPOPUP);

// SETTING SHORTCUT KEYS OF MENUS IN THE MAIN
// MENUBAR
FILE.setMnemonic(KeyEvent.VK_F);
EDIT.setMnemonic(KeyEvent.VK_E);
SEARCH.setMnemonic(KeyEvent.VK_S);
HELP.setMnemonic(KeyEvent.VK_H);
```

```
// SETTING SHORTCUT KEYS OF MENUITEMS IN THE FILE MENU
    NEWFILE.setMnemonic(KeyEvent.VK_N);
    OPENFILE.setMnemonic(KeyEvent.VK_O);
    SAVEFILE.setMnemonic(KeyEvent.VK_S);
    SAVEASFILE.setMnemonic(KeyEvent.VK_A);
    EXITFILE.setMnemonic(KeyEvent.VK_X);

// SETTING SHORTCUT KEYS OF MENUITEMS IN THE EDIT MENU
    UNDOEDIT.setMnemonic(KeyEvent.VK_U);
    CUTEDIT.setMnemonic(KeyEvent.VK_T);
    COPYEDIT.setMnemonic(KeyEvent.VK_C);
    PASTEDIT.setMnemonic(KeyEvent.VK_P);
    DELETEDIT.setMnemonic(KeyEvent.VK_L);
    SELECTEDIT.setMnemonic(KeyEvent.VK_A);
    TIMEDIT.setMnemonic(KeyEvent.VK_D);
    WORDEDIT.setMnemonic(KeyEvent.VK_W);
    FONTEDIT.setMnemonic(KeyEvent.VK_F);

// SETTING SHORTCUT KEYS OF MENUITEMS IN THE SEARCH MENU
    FINDSEARCH.setMnemonic(KeyEvent.VK_F);
    FINDNEXTSEARCH.setMnemonic(KeyEvent.VK_N);
```

```
// SETTING SHORTCUT KEYS OF MENUITEM IN THE HELP MENU
    ABOUTHELP.setMnemonic(KeyEvent.VK_A);

// SETTING SHORTCUT KEYS OF MENUITEMS IN THE POPUPMENU
    UNDOPOPUP.setMnemonic(KeyEvent.VK_U);
    CUTPOPUP.setMnemonic(KeyEvent.VK_T);
    COPYPOPUP.setMnemonic(KeyEvent.VK_C);
    PASTEPOPUP.setMnemonic(KeyEvent.VK_P);
    DELETEPOPUP.setMnemonic(KeyEvent.VK_D);
    SELECTPOPUP.setMnemonic(KeyEvent.VK_A);

// SETTING ACCELERATOR KEYS OF SOME MENUITEMS IN
   THE EDIT MENU
    UNDOEDIT.setAccelerator(KeyStroke.getKeyStroke(
    KeyEvent.VK_Z, ActionEvent.CTRL_MASK));
    CUTEDIT.setAccelerator(KeyStroke.getKeyStroke(
    KeyEvent.VK_X, ActionEvent.CTRL_MASK));
    COPYEDIT.setAccelerator(KeyStroke.getKeyStroke(
    KeyEvent.VK_C, ActionEvent.CTRL_MASK));
    PASTEDIT.setAccelerator(KeyStroke.getKeyStroke(
    KeyEvent.VK_V, ActionEvent.CTRL_MASK));
```

ADDING LISTENER **INSIDE CONSTRUCTOR OF MAIN CLASS**

```
// ADDING LISTENERS TO THE MENUITEMS IN FILE MENU
NEWFILE.addActionListener(this);
OPENFILE.addActionListener(this);
SAVEFILE.addActionListener(this);
SAVEASFILE.addActionListener(this);
EXITFILE.addActionListener(this);

// ADDING LISTENERS TO THE MENUITEMS IN EDIT MENU
//FIRST ONE IS FOR UNDO
text.getDocument().addUndoableEditListener(new
UndoListener());
UNDOEDIT.addActionListener(new UndoAction());
CUTEDIT.addActionListener(this);
COPYEDIT.addActionListener(this);
PASTEDIT.addActionListener(this);
DELETEDIT.addActionListener(this);
SELECTEDIT.addActionListener(this);
TIMEDIT.addActionListener(this);
WORDEDIT.addActionListener(this);
FONTEDIT.addActionListener(this);

// ADDING LISTENERS TO THE MENUITEMS IN SEARCH MENU
FINDSEARCH.addActionListener(this);
FINDNEXTSEARCH.addActionListener(this);

// ADDING LISTENERS TO THE MENUITEM IN HELP MENU
ABOUTHELP.addActionListener(this);
```

```
// ADDING LISTENERS TO THE MENUITEMS IN POPUPMENU
UNDOPOPUP.addActionListener(new UndoAction());
CUTPOPUP.addActionListener(this);
COPYPOPUP.addActionListener(this);
PASTEPOPUP.addActionListener(this);
DELETEPOPUP.addActionListener(this);
SELECTPOPUP.addActionListener(this);

// ADDING MOUSELISTENER TO RIGHT CLICK FOR THE
POPUPLISTENER
text.addMouseListener(new MouseAdapter() {
    public void mousePressed(MouseEvent e) {
        if (e.isPopupTrigger()) {
            POPUP.show(e.getComponent(), e.getX(),
            e.getY());
        }

    }

    public void mouseReleased(MouseEvent e) {
        if (e.isPopupTrigger()) {
            POPUP.show(e.getComponent(), e.getX(),
            e.getY());
        }
    }
});
```

```
    // ADDING WINDOWLISTENER TO HANDLE CLOSE
    // WINDOW EVENT
        addWindowListener(new WindowAdapter() {
          public void windowClosing(WindowEvent e)
           {
                exitApln();
           }
        });
    }
```

END OF CONSTRUCTOR OF

EVENT HANDLING **INSIDE OF MAIN CLASS**

```
// HANDLING ALL EVENTS OF THE TEXT EDITOR
public void actionPerformed(ActionEvent e) {

    // ACTION FOR NEW MENU OPTION OF FILE MENU
    if (e.getSource() == NEWFILE) {
        newfile();
    }

    // ACTION FOR OPEN MENU OPTION OF FILE MENU
    if (e.getSource() == OPENFILE) {
        open();
    }

    // ACTION FOR SAVE MENU OPTION OF FILE MENU
    if (e.getSource() == SAVEFILE) {
        save();
    }

    // ACTION FOR SAVEAS MENU OPTION OF FILE MENU
    if (e.getSource() == SAVEASFILE) {
        opened = false;
        save();
    }

    // ACTION FOR EXIT MENU OPTION OF FILE MENU
    if (e.getSource() == EXITFILE) {
        exitApln();
    }
```

```
    // ACTION FOR UNDO MENU OPTION OF EDIT MENU AND
    // POPUPMENU
    if ((e.getSource() == UNDOEDIT) ||
    (e.getSource() == UNDOPOPUP)) {
        new UndoAction();
    }

    // ACTION FOR CUT MENU OPTION OF EDIT MENU AND
    // POPUPMENU
    if ((e.getSource() == CUTEDIT) ||
    (e.getSource() == CUTPOPUP)) {
        text.cut();
    }

    // ACTION FOR COPY MENU OPTION OF EDIT MENU AND
    // POPUPMENU
    if ((e.getSource() == COPYEDIT) ||
    (e.getSource() == COPYPOPUP)) {
        text.copy();
    }
```

```
    // ACTION FOR PASTE MENU OPTION OF EDIT MENU
    AND POPUPMENU
    if ((e.getSource() == PASTEDIT) ||
    (e.getSource() == PASTEPOPUP)) {
        text.paste();
    }

    // ACTION FOR DELETE MENU OPTION OF EDIT MENU
    AND POPUPMENU
    if ((e.getSource() == DELETEDIT) ||
    (e.getSource() == DELETEPOPUP)) {
        text.replaceSelection(null);
    }
    // ACTION FOR SELECTALL MENU OPTION OF EDIT
    MENU AND POPUPMENU
    if ((e.getSource() == SELECTEDIT) ||
    (e.getSource() == SELECTPOPUP)) {
        text.selectAll();
    }
    // ACTION FOR TIME/DATE MENU OPTION OF
    EDIT MENU
    if (e.getSource() == TIMEDIT) {
        Date currDate;
        String dd;
        currDate = new java.util.Date();
        dd = currDate.toString();
        text.insert(dd, text.getCaretPosition());
    }
```

```
    // ACTION FOR WORD WRAP MENU OPTION OF EDIT MENU
    if (e.getSource() == WORDEDIT) {
        if (WORDEDIT.isSelected())
            text.setLineWrap(true);
        else
            text.setLineWrap(false);
    }

    // ACTION FOR SET FONT MENU OPTION OF EDIT MENU
    if (e.getSource() == FONTEDIT) {
        new fontDialogBox();
    }

    // ACTION FOR FIND MENU OPTION OF SEARCH MENU
    if (e.getSource() == FINDSEARCH) {
        wholeText = text.getText();
        findString = JOptionPane.showInputDialog(null,
        "Find What", "Find",
        JOptionPane.INFORMATION_MESSAGE);

        ind = wholeText.indexOf(findString, 0);
        text.setCaretPosition(ind);
        text.setSelectionStart(ind);
        text.setSelectionEnd(ind + findString.length());
    }
```

```
    // ACTION FOR FIND NEXT MENU OPTION OF SEARCH
    MENU
    if (e.getSource() == FINDNEXTSEARCH) {
        wholeText = text.getText();
        findString =
        JOptionPane.showInputDialog(null,
        "Find What", "Find Next",
        JOptionPane.INFORMATION_MESSAGE);
        ind = wholeText.indexOf(findString,
        text.getCaretPosition());
        text.setCaretPosition(ind);
        text.setSelectionStart(ind);
        text.setSelectionEnd(ind +
        findString.length());
    }

    // ACTION FOR ABOUT MENU OPTION OF HELP MENU
    if (e.getSource() == ABOUTHELP) {
    JOptionPane.showMessageDialog(null,
    "This is a simple Text Editor application
    built using Java.", "About Editor",
    JOptionPane.INFORMATION_MESSAGE);
    }
}
```

END OF EVENT HANDLING

CREATING NEW FILE **INSIDE OF MAIN CLASS**

```
// ACTION FOR NEW MENU OPTION OF FILE MENU
public void newfile() {
    if (!text.getText().equals("")) {
        opened = false;
        int confirm = JOptionPane.showConfirmDialog(null,
                    "Text in the Untitled file has
                    changed. \n Do you want to save the
                    changes?","New File",
                    JOptionPane.YES_NO_CANCEL_OPTION,
                    JOptionPane.INFORMATION_MESSAGE);

        if (confirm == JOptionPane.YES_OPTION) {
           save();
           text.setText(null);
        } else if (confirm == JOptionPane.NO_OPTION) {
            text.setText(null);
        }
    }
}
```

OPENING | **INSIDE OF MAIN**

```
// ACTION FOR OPEN MENU OPTION OF FILE MENU
public void open() {
    text.setText(null);
    JFileChooser ch = new JFileChooser();
    ch.setCurrentDirectory(new File("."));
    ch.setFileFilter(new
    javax.swing.filechooser.FileFilter() {
        public boolean accept(File f) {
        return f.isDirectory() ||
     f.getName().toLowerCase().endsWith(".java");
        }

        public String getDescription() {
            return "Java files";
        }
    });
    int result = ch.showOpenDialog(new JPanel());
    if (result == JFileChooser.APPROVE_OPTION) {
        filename =
        String.valueOf(ch.getSelectedFile());
        setTitle(filename);
        opened = true;
        FileReader fr;
        BufferedReader br;
```

```
        try {
            fr = new FileReader(filename);
            br = new BufferedReader(fr);
            String s;
            while ((s = br.readLine()) != null) {
                text.append(s);
                text.append("\n");
            }
            fr.close();
        } catch (FileNotFoundException ex) {
            JOptionPane.showMessageDialog(this,
            "Requested file not found", "Error Dialog
            box", JOptionPane.ERROR_MESSAGE);
        } catch (Exception ex) {
            System.out.println(ex);
        }
    }
}
```

SAVING FILE **INSIDE OF MAIN CLASS**

```
// ACTION FOR SAVE MENU OPTION OF FILE MENU
public void save() {
    if (opened == true) {
        try {
            FileWriter f1 = new
            FileWriter(filename);
            f1.write(text.getText());
            f1.close();
            opened = true;
        } catch (FileNotFoundException ex) {
          JOptionPane.showMessageDialog(this,
          "Requested file not found", "Error
          Dialog box",
          JOptionPane.ERROR_MESSAGE);
        } catch (IOException ioe) {
            ioe.printStackTrace();
        }
    } else {
        JFileChooser fc = new JFileChooser();
        fc.setCurrentDirectory(new File("."));
        int result = fc.showSaveDialog(new
        JPanel());
        if (result ==
        JFileChooser.APPROVE_OPTION) {
            filename =
            String.valueOf(fc.getSelectedFile());
            setTitle(filename);
```

```
            try {
                FileWriter f1 = new
                FileWriter(filename);
                f1.write(text.getText());
                f1.close();
                opened = true;
            } catch (FileNotFoundException ex) {
                JOptionPane.showMessageDialog(this,
                "Requested file not found", "Error Dialog
                box", JOptionPane.ERROR_MESSAGE);
            }

            catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }

    }
}
```

EXIT PROGRAM **INSIDE OF MAIN CLASS**

```
// ACTION FOR EXIT MENU OPTION OF FILE MENU AND CLOSE
WINDOW BUTTON
public void exitApln() {
    if (!text.getText().equals("")) {
        int confirm = JOptionPane.showConfirmDialog(null,
        "Text in the file has changed. \n Do you want to
        save the changes?","Exit",
        JOptionPane.YES_NO_CANCEL_OPTION,
        JOptionPane.INFORMATION_MESSAGE);

        if (confirm == JOptionPane.YES_OPTION) {
            save();
            dispose();
            System.exit(0);
        }else if (confirm == JOptionPane.CANCEL_OPTION) {
            e = new TextEditer();
            String s = text.getText();
            e.setVisible(true);
            e.text.setText(s);
        } else if (confirm == JOptionPane.NO_OPTION) {
            dispose();
            System.exit(0);
        }
    } else {
        System.exit(0);
    }
}
```

HANDLING UNDO MENU **INSIDE OF MAIN CLASS**

```
// CLASS FOR HANDLING UNDO MENU OPTION OF EDIT AND POPUP
MENU
class UndoAction extends AbstractAction {

    public void actionPerformed(ActionEvent e) {
        try {
            undo.undo();
        } catch (CannotUndoException ex) {
            System.out.println("Unable to undo: " + ex);
            ex.printStackTrace();
        }
        update();
    }

    protected void update() {
        if (undo.canUndo()) {
            setEnabled(true);
            putValue("Undo",
             undo.getUndoPresentationName());
        } else {
            setEnabled(false);
            putValue(Action.NAME, "Undo");
        }
    }
}
```

ADDING LISTENER TO UNDOMENU | **INSIDE OF MAIN CLASS**

```
// CLASS FOR UNDOLISTENER
class UndoListener implements UndoableEditListener {
    public void undoableEditHappened(UndoableEditEvent e){
        undo.addEdit(e.getEdit());
        undoAction.update();
    }
}
```

FontDialogBox Class **INSIDE OF MAIN CLASS**

```
// CLASS FOR BUILDING AND DISPLAYING FONT DIALOG BOX
class fontDialogBox extends JFrame implements
ActionListener {
    // DECLARATION OF ALL VARIABLES USED IN fontDialogBox
CLASS

    String availableFontString[] =
    GraphicsEnvironment.getLocalGraphicsEnvironment().
    getAvailableFontFamilyNames();
    JList<String> fontList = new
    JList<String>(availableFontString);
    JLabel fontLabel = new JLabel("Font");
    JTextField valueFont = new JTextField("Arial");
    JScrollPane fontPane = new JScrollPane(fontList);

    String fontStyleString[] = { "Normal", "Bold",
                            "Italic", "Bold Italic" };
    JList<String> styleList = new
    JList<String>(fontStyleString);
    JLabel styleLabel = new JLabel("Style");
    int v = ScrollPaneConstants.
                        VERTICAL_SCROLLBAR_ALWAYS;
    int h = ScrollPaneConstants.
                    HORIZONTAL_SCROLLBAR_AS_NEEDED;
    JScrollPane stylePane = new JScrollPane(styleList, v,
                                                    h);
    JTextField valueStyle = new JTextField("Normal");
```

```
String fontSizeString[] = { "8", "10", "12", "14",
            "16", "18", "20", "22", "24", "28" };
JList<String> sizeList = new
JList<String>(fontSizeString);
JLabel sizeLabel = new JLabel("Font size");
JScrollPane sizePane = new JScrollPane(sizeList);
JTextField valueSize = new JTextField("12");

JButton okButton = new JButton("OK");
JButton cancelButton = new JButton("Cancel");

JLabel sampleLabel = new JLabel("Sample:");
JTextField sample = new JTextField("   AaBbCc");

Font selectedFont;
```

FontDialogBox Constructor | **INSIDE OF FontDialogBox CLASS**

```
// DEFAULT CONSTRUCTOR OF fontDialogBox CLASS
public fontDialogBox() {
    setSize(500, 300);
    setTitle("Font");
    setVisible(true);
    sample.setEditable(false);

    getContentPane().setLayout(null);

    fontLabel.setBounds(10, 10, 170, 20);
    valueFont.setBounds(10, 35, 170, 20);
    fontPane.setBounds(10, 60, 170, 150);

    styleLabel.setBounds(200, 10, 100, 20);
    valueStyle.setBounds(200, 35, 100, 20);
    stylePane.setBounds(200, 60, 100, 150);

    sizeLabel.setBounds(320, 10, 50, 20);
    valueSize.setBounds(320, 35, 50, 20);
    sizePane.setBounds(320, 60, 50, 150);

    okButton.setBounds(400, 35, 80, 20);
    cancelButton.setBounds(400, 60, 80, 20);

    sampleLabel.setBounds(150, 235, 50, 30);
    sample.setBounds(200, 235, 100, 30);
```

```
    getContentPane().add(fontLabel);
    getContentPane().add(fontPane);
    getContentPane().add(valueFont);

    getContentPane().add(styleLabel);
    getContentPane().add(stylePane);
    getContentPane().add(valueStyle);

    getContentPane().add(sizeLabel);
    getContentPane().add(sizePane);
    getContentPane().add(valueSize);

    getContentPane().add(okButton);
    getContentPane().add(cancelButton);
    getContentPane().add(sampleLabel);
    getContentPane().add(sample);

    okButton.addActionListener(this);
    cancelButton.addActionListener(this);
```

```
fontList.addListSelectionListener(new
ListSelectionListener() {
    public void valueChanged(ListSelectionEvent event) {
        if (!event.getValueIsAdjusting()) {
           valueFont.setText(fontList.getSelectedValue().
           toString());
           selectedFont = new Font(valueFont.getText(),
           styleList.getSelectedIndex(),
           Integer.parseInt(valueSize.getText()));
           sample.setFont(selectedFont);
        }
    }
});

styleList.addListSelectionListener(new
ListSelectionListener() {
    public void valueChanged(ListSelectionEvent event) {
       if (!event.getValueIsAdjusting()) {
         valueStyle.setText(styleList.getSelectedValue().
         toString());
         selectedFont = new Font(valueFont.getText(),
         styleList.getSelectedIndex(),
         Integer.parseInt(valueSize.getText()));
         sample.setFont(selectedFont);
       }
    }
});
```

```
    sizeList.addListSelectionListener(new
    ListSelectionListener() {
       public void valueChanged(ListSelectionEvent event)
       {
          if (!event.getValueIsAdjusting()) {
            valueSize.setText(sizeList.getSelectedValue().
            toString());
            selectedFont = new Font(valueFont.getText(),
            styleList.getSelectedIndex(),
            Integer.parseInt(valueSize.getText()));
            sample.setFont(selectedFont);
          }
        }
    });
}// END OF DEFAULT CONSTRUCTOR OF fontdialogBox CLASS
```

END OF FontDialogBox Constructor

FontDialogBox EVENT HANDLING | **INSIDE OF FontDialogBox CLASS**

```
    public void actionPerformed(ActionEvent ae) {
        if (ae.getSource() == okButton) {
            selectedFont = new Font(valueFont.getText(),
            styleList.getSelectedIndex(),
            Integer.parseInt(valueSize.getText()));
            text.setFont(selectedFont);
            setVisible(false);
        }
        if (ae.getSource() == cancelButton) {
                setVisible(false);
        }
    }
}// END OF fontDialogBox CLASS
```

END OF FontDialogBox CLASS

MAIN METHOD

INSIDE OF MAIN CLASS

```
    // MAIN FUNCTION OF TextEditer CLASS
    public static void main(String args[]) {
        new TextEditer();
    }
}// END OF TextEditer CLASS
```

END OF MAIN CLASS

Testing

In the software development life cycle, after the requirement analysis, feasibility study, design and coding phase, one of the phases of utmost importance is the testing phase. In this phase, we get to see if the expected and final outcomes are same with all the required specifications being maintained as per the requirement. This is the debugging phase of the application on its entirety and not simple debugging of few lines of code on the IDE.

This phase starts with the smallest unit of a code through Unit Testing and travels a long way till it ends with User Acceptance Testing. For industrial applications, testing is performed through highly automated tools like Docker for

NodeJS applications, but such automated testing tools are out of scope for this project. This testing phase helps in finding bugs which can be at the code level, system level, environment level, then fixing those bugs, then retesting the same functionalities to check the new changes are compatible with others and then at the end of many testing strategies the product is delegated to corresponding authority for release. Testing at the application/code level is performed by writing test cases.
When code is nothing but human typed language with a sense of grammar and context understandable by the OS and computer scientists, there can always exist errors. We can eliminate these errors by thinking about corner cases or scenarios that might never happen but if happens, our application is strong enough to handle them. Testing the application with every possible

scenario which it can/cannot handle and still our application staying steadfast is what we target for.

Testing Levels

✞ Integration Testing

Once each individual part of the system is tested, every smallest unit is tested, different modules of the system are now integrated together and tested. Whether the integration works or whether a part of the system that is functional individually starts failing when integrated with another part is what integration testing is all about.

✞ System Testing

That an integrated system meets all its specifications and requirements is decided by system Testing.
Regression Testing – Once the system is debugged, it is tested again to see if it is compatible with the changes made and compatible with any changes made to the environment.

✞ Performance Testing

Testing how the system performs like slow/fast and how it performs under certain workloads.

Tests performed on User Module

MODULE	TEST CASE	EXPECTED RESULT	OBSERVED RESULT
User	Creating a new file	New file created	PASS
User	Opening an existing file	Existing file open	PASS
User	Saving a file	File saved	PASS
User	Saving a file in desired folder	File saved	PASS
User	Exit from application	Exited	PASS
User	Cutting the text	Text copied and removed	PASS
User	Copying the text	Text Copied	PASS

User	Getting The previous text	Text undo	PASS
User	Deleting the required text	Text Deleted	PASS
User	Selecting the all text	All texts are selected	PASS
User	Wrapping the word	Word wrapped	PASS
User	Setting the font of text	Font changed	PASS
User	Searching the specific text	Text finds	PASS
User	Moving on next options	Next option finds	PASS
User	Knowing about editor	Describe to editor	PASS

User	Getting the current time and date	Show current time and date	PASS

✞ Testing Principles

- All tests should be traceable to end user requirements
- Tests should be planned long before testing begins • Testing should begin on a small scale and progress towards testing in large
- Exhaustive testing is not possible
- To be most effective testing should be conducted by a independent third party.

The primary objective for test case design is to derive a set of tests that has the highest livelihood for uncovering defects in software. To accomplish this objective two different categories of test case design techniques are used. They are

- **White box testing**
- **Black box testing**

○White-box testing

White box testing focus on the program control structure. Test cases are derived to ensure that all statements in the program have been executed at least once during testing and that all logical conditions have been executed.

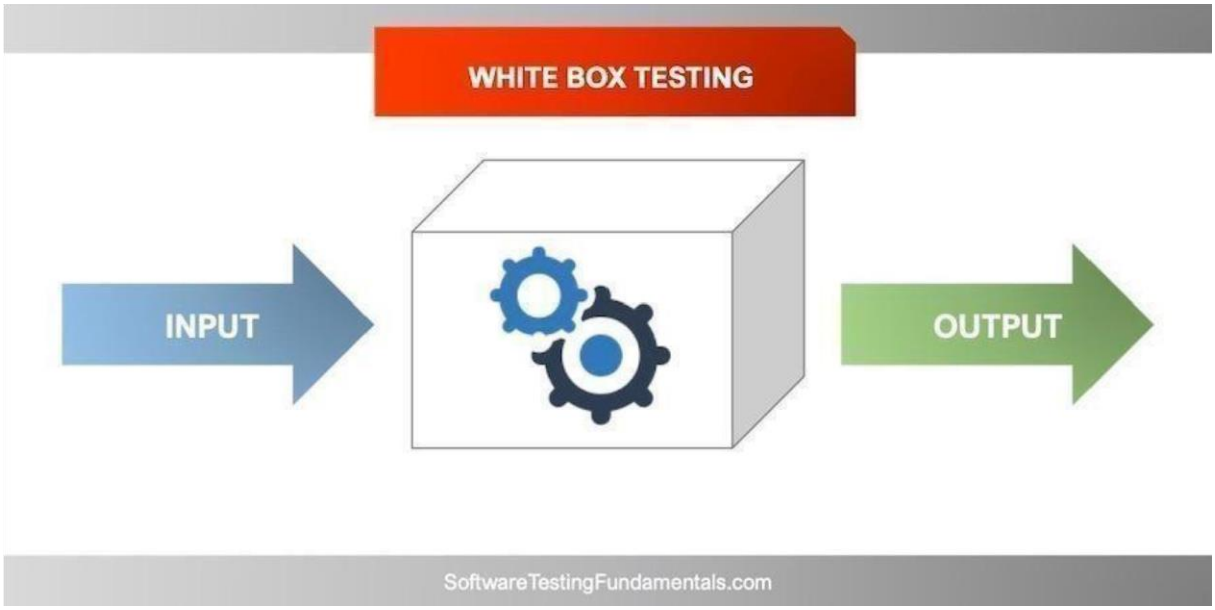


fig13 - White Box Testing

Black-box testing

Black box testing is designed to validate functional requirements without regard to the internal workings of a program. Black box testing mainly focuses on the information domain of the software, deriving test cases by partitioning input and output in a manner that provides through test coverage. Incorrect and missing

functions, interface errors, errors in data structures, error in functional logic are the errors falling in this category.

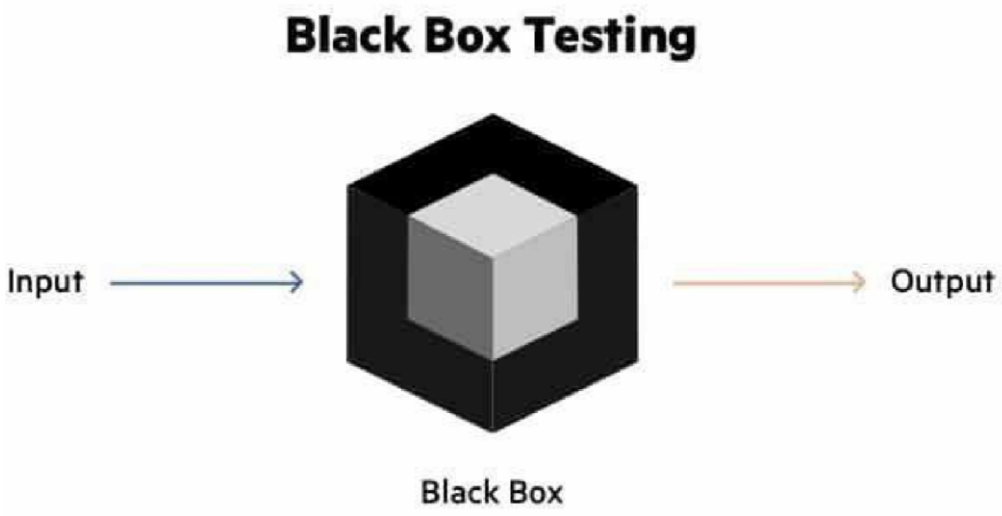


fig14 - Black Box Testing

Bugs Encountered

While I performed initial testing on this application, I have faced few bugs and have fixed them. In the Text editor application, initially I have not made proper code for opening an existing file. Due to this, when user wanted to open an existing file then new window was opening. We fixed this bug by proper coding and the functionality worked perfect.

While testing the text font in edit section, a bug appeared where if we want to set the font size, font style and font

together, then random one used to work. We fixed this bug by add condition of them and functionality worked perfect that help user to working on text formatting very easy manner.

While testing the cut, copy and paste operation on edit section of text editor application, a bug appeared. This bug was more confusing for us because by minor mistake on coding section, sometime copy was performing cut operation and sometime copy. This was very disturbing bug. We fixed this bug by go through the code and make required changes on that.

I also did some functional enhancements which helped improve our editor quality too.
By texting all possible condition and solving all bugs, our project look like perfect and working smooth and fast.

Implementation

The Text Editor has one Java file containing all the code. To run the Java file, we need to install jdk 7 or higher version on window 7 or higher system.

For downloading jdk 7 or higher we can visit oracle website (https://download.oracle.com/java/19/latest/jdk-19_windows-x64_bin.exe).

After Installing jdk & or higher we can now compile and run our Java file. For compile and running Java file there are two ways:

1. By coping the file in the bin folder of the installed Java jkd which is by default (C:\Program Files\Java\jdk-18.0.2.1\bin) If no change is made during installation of jdk. After copying the Java code file in bin we need to open command prompt. Command prompt must have the permission of creating file if Java jdk folder exist in C drive in newer windows. If the permission is not allowed then we need to copy java jdk folder in another directory.

   Now, once all the above tasks are performed, then we need to follow the following steps:

Step 1: Navigate to the directory in which bin of jdk is created or copied using cd command of command prompt.

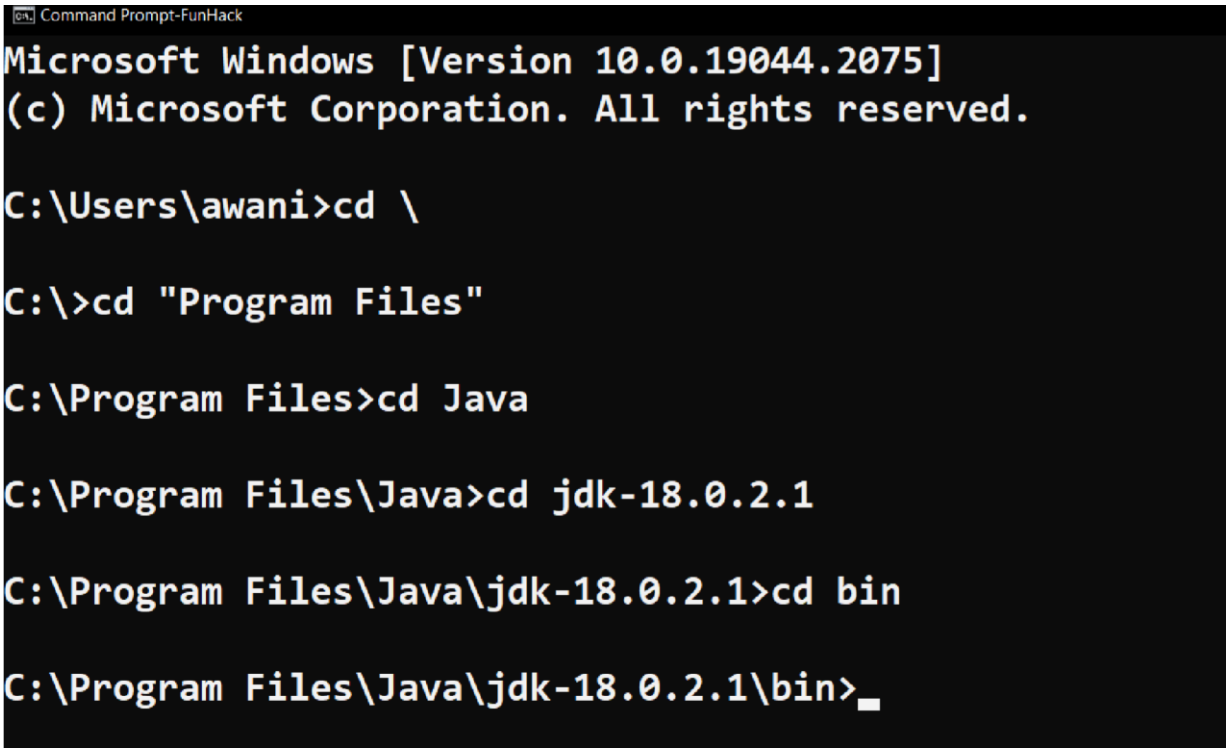


fig15 - Navigation of Directory

Step 2: After putting prompt directory in Java jdk bin we write following code to compile java code file.

prompt directory>javac TextEditor

After successful compilation several files will be created in which one is TextEditor.class which we use in future steps.

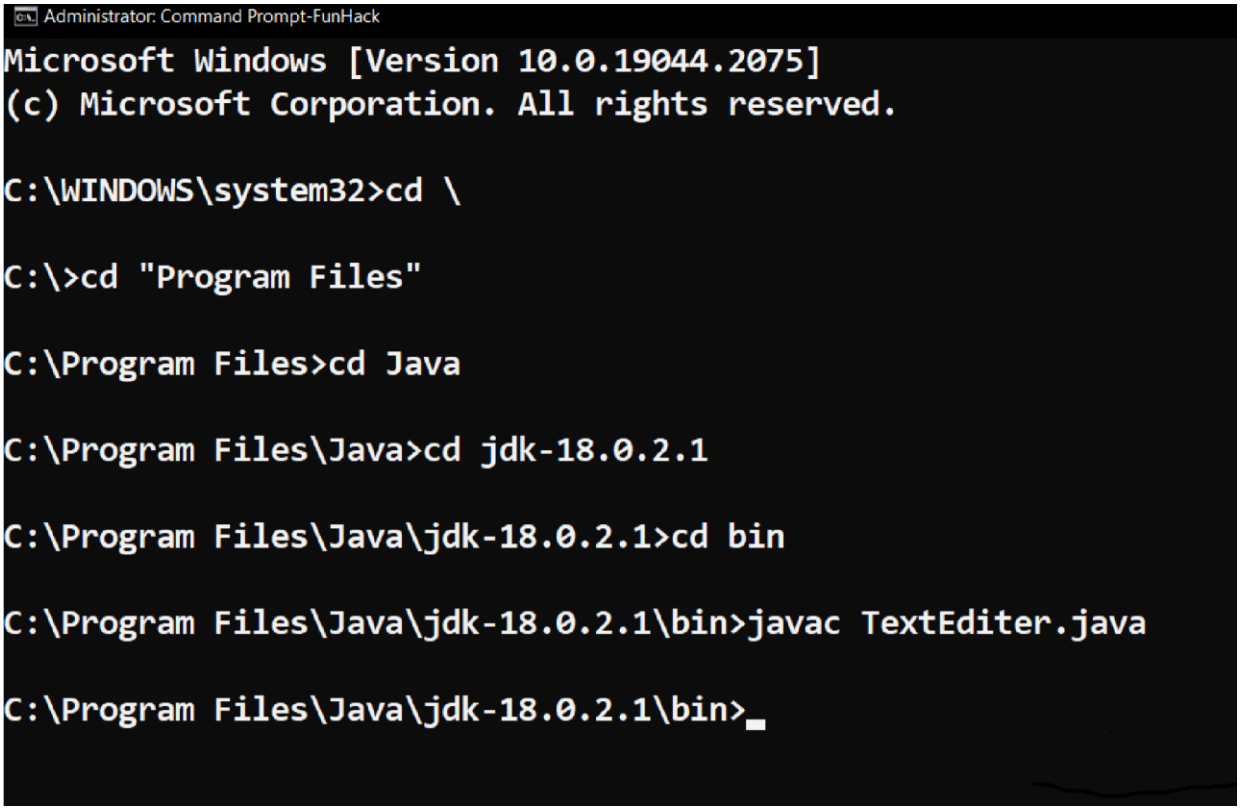


fig16 - Compilation of Code

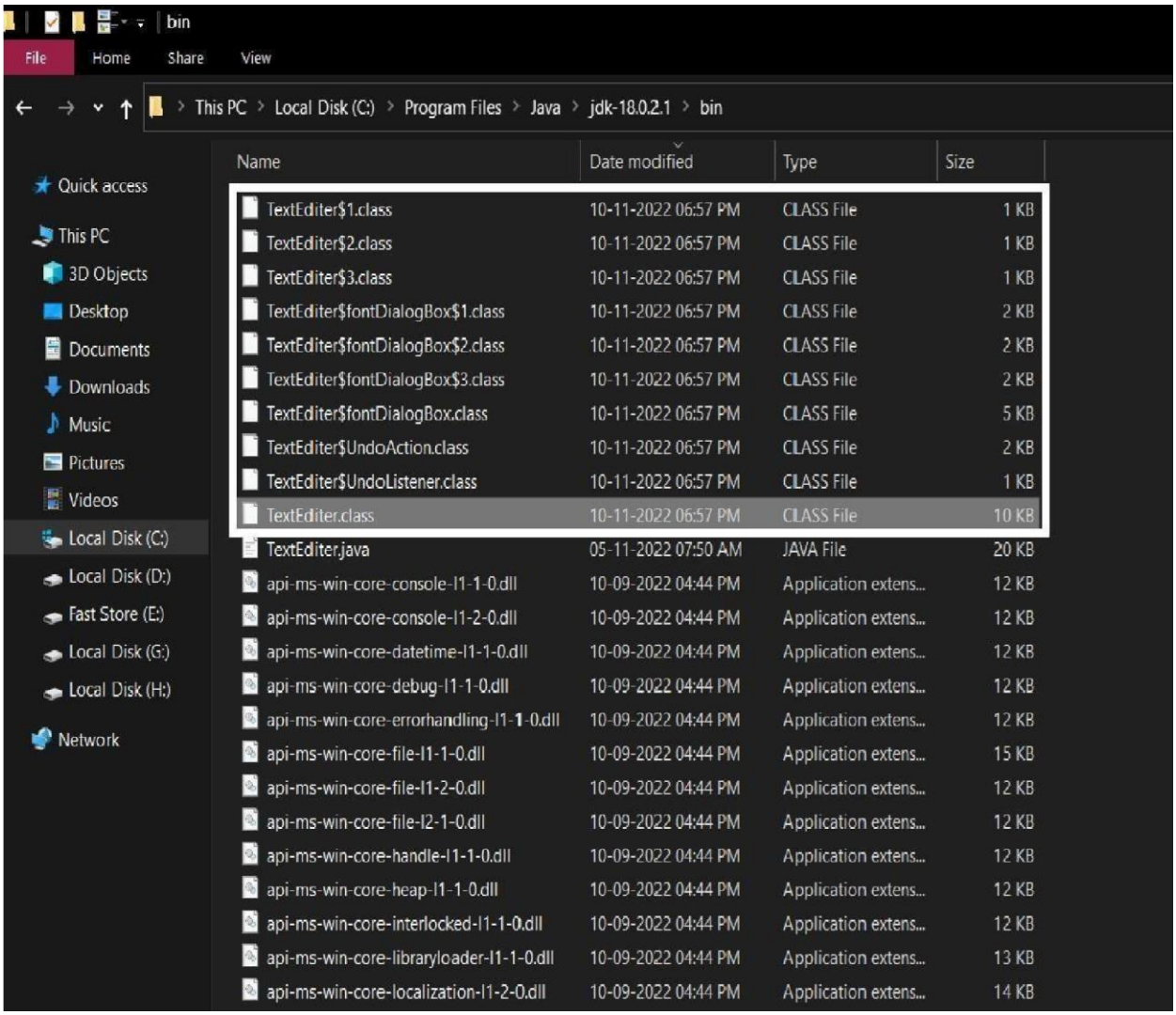


fig17 - File created on compilation

Step 3: After successful compilation and creation of TextEditor.class file we will run the code or TextEditor finally by writing following code:

prompt directory>java TextEditor

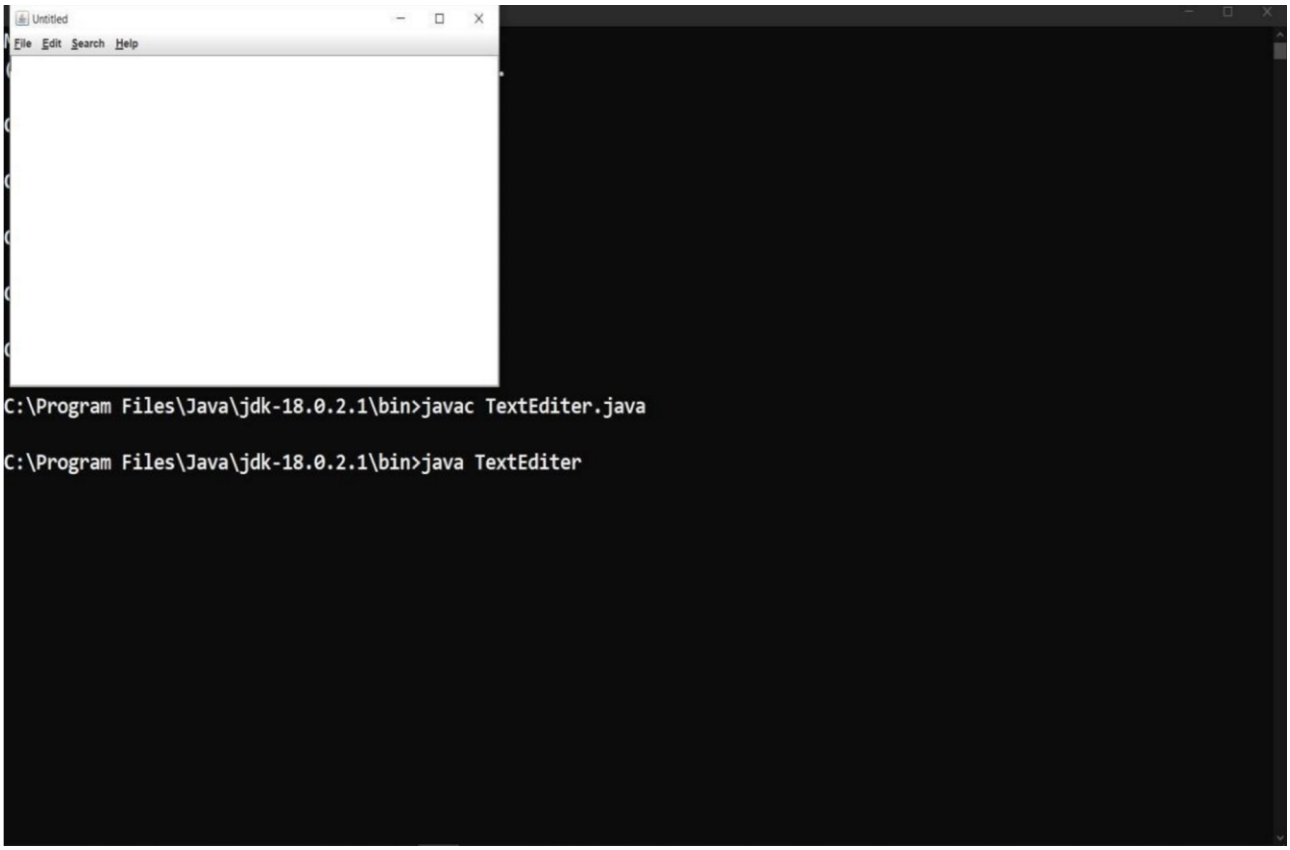


fig18 - Running of Code

Step 4: Now, a GUI window of TextEditor appears and it is ready to be used.

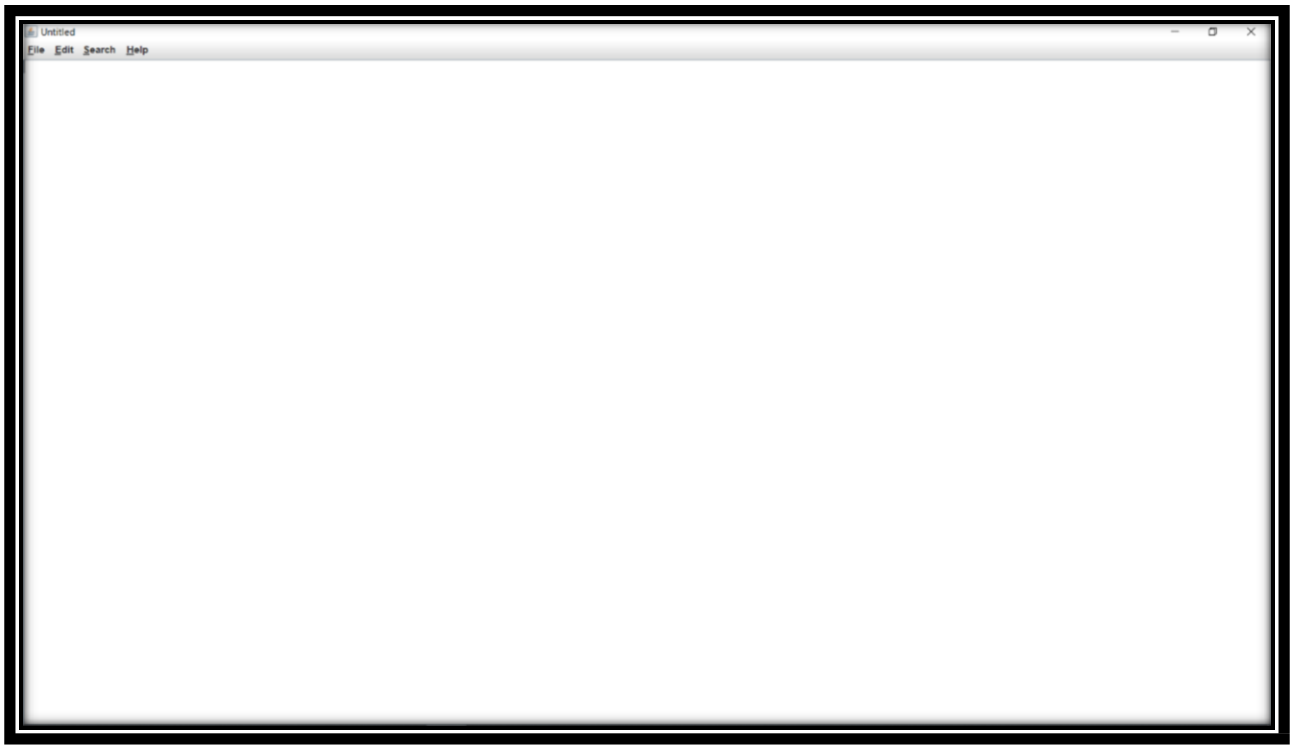


fig19 - Text Editor

2. Second way of running the java code file is by saving the path of java jdk bin folder is environment variable path. We can do it as following:

Step 1: Go to settings
Step 2: Click on system settings
Step 3: Click on About
Step 4: Click on Advanced System
Step 5: Click on Environment Variables

Step 6: Select path from system variables

Step 7: Click on Edit

Step 8: Click on New

Step 9: Write or paste the path of java jdk bin folder.

Step 10: Click on Ok

Step 11: Click on Ok again

Step 12: Click on Ok again

Now, Environment Variable is saved.

Now, we can run java code from any folder of the system. Now, open Command prompt and follow the same steps that are explained in the previous way to run Text Editor.

Limitation

Text Editor is small software with less feature and many benefits one of them is a smaller number of limitations according to requirement.

We know more functions make program complex and slow, we want to make it fast so we provide only limited number of features, some of high-cost processor features are not added like color changing of text and background. Changing color of text and background uses too much processor to display it not screen so it is a missing feature from our text editor.

At any point program crashes due to any bug encountered date loss may be occurred. We try our best to save data of user but at some point, due to interruption from operating system or any third-party app, data may be lost by the application.

In this test editor we don't provide any feature for saving last settings or remembering the format of previously opened files. So, Text Editor will always open in default state.

These are some limitations of our application. By the way limitation is a subject of user’s interest, what user wants and what they don't. So, they may feel difference in limitations of this application."

Conclusion

Text Editor is one of the most used applications for writing and editing text. Text editor popularity attract different developers to develop unique and creative text editors with lots of attractive features.

We also try to develop our Text Editor in this project with minimum features which is enough for editing or writing normal text files and saving them for furthers uses. Throughout this project we learnt a lot of things like basis of programming, how to create GUI application, how to use different libraries in application and a lot more about programs. Also, while creating documentation of project, we learnt how to represent our project, how to represent its DFD's or ER-Diagram etc. Overall, it's a good experience to work with this project.


THANKYOU

-