A Mini Project Report

On

**FILE COMPRESSION**

Submitted in partial fulfilment of the

Requirements for the award of degree of

**Bachelor of Engineering**

**In**

**Computer Science and Engineering**

By

Roll no: Name

1608-14-733-054: Ashwin Ingle

1608-14-733-045: V.Abhishek

1608-14-733-039: K.Siddharth

Under the guidance of

Mrs J Samatha, Asst.prof CSE Dept



**Department of Computer Science and Engineering**

**Matrusri Engineering College**

**(Affiliated to OU, Approved by AICTE)**

**Saidabad, Hyderabad-500059.**

**(2012-2016)**

**Department of Computer Science and Engineering**

**Matrusri Engineering College**

**(Affiliated to OU, Approved by AICTE)**

**Saidabad, Hyderabad-500059.**

**(2012-2016)**



**CERTIFICATE**

This is to certify that the mini project entitled “**FILE COMPRESSION**” being submitted by  **Ashwin Ingle, V. Abhishek, K. Siddharth** bearing the roll number **1608-14-733-054,1608-14-733-045,1608-14-733-039** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Engineering** in **Computer Science and Engineering** to **MATRUSRI ENGINEERING COLLEGE.** Affiliated to OU, is a record of bonafide work carried out by us under the guidance and supervision from July-2015 to April-2016.

**Mrs J. Samatha Prof.G.Veerashalingam**

**(Internal Guide) (HOD, CSE dept).**

**ACKNOWLEDGEMENT**

This project report is the outcome of the efforts of many people, who have driven my passion to explore into Advance Java and AWTs. We have received great guidance, encouragement and support from them and have learned a lot because of their willingness to share their knowledge and experience.

I express my sincere gratitude to Prof. G.Veereshalingam, Department of Computer Science and Engineering, Matrusri Engineering College, for his precious suggestions, motivation and co-operation for the successful completion of this project work.

It is my privilege and pleasure to express my profound sense of respect, gratitude and indebtedness to my guide Mrs.J.Samatha, Assistant professor, Department of Computer Science and Engineering, Matrusri Engineering College, for her indefatigable inspiration, guidance, cogent discussion, constructive criticisms and encouragement throughout this dissertation work.

We also thank our Mini Project Coordinator Ms.B.J.Praveena ,Assistant Professor, Department of CSE .

Last but not the least, we wish to acknowledge our friends, family members and colleagues for giving moral strength and helping me to complete this dissertation.

**Name: Ashwin Ingle, V.Abhishek, K.Siddharth.**

**Roll no: 1608-14-733-054,**

**: 1608-14-733-045,**

**: 1608-14-733-039,**

**Dept. of Computer Science & Engineering,**

**Matrusri Engineering College.**

**DECLARATION**

We hereby declare that the mini project entitled “**FILE COMPRESSION**” submitted to the Osmania University in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science Engineering, is a bonafied of an original work done by me under the guidance of Mrs J.Samatha Assistant Professor and this project work have not been submitted to any other university for the award of any degree or diploma.

**Ashwin Ingle, V. Abhishek, K.Siddharth**

**1608-14-733-054,**

**1608-14-733-045,**

**1608-14-733-039.**

**Date: 22 October 2016**

**ABSTRACT**

The Domain “GZip” lets one to reduce the overall number of bits and bytes in a file so it can be transmitted faster over slower Internet connections, or take up less space on a disk. The user need not depend on third party software’s like WinZip, WinRAR. The main algorithm is Huffman Algorithm for Compression/Decompression. Huffman coding is an encoding algorithm for lossless data compression. Huffman coding uses a specific method for choosing the representation for each symbol, resulting in a prefix code (that is, the bit string representing some symbol is never a prefix of the bit string representing any other symbol) that expresses the most common characters using shorter strings of bits than are used for less common source symbols.

**table of contents**

**S.no contents page no.**

1. Introduction 7

1.1. Problem Definition 7

1.2. Objectives 7

1.3. Existing System 7

1.4. Proposed System 8

1. Literature Survey 9
2. Requirements Analysis 11

3.1. Object Oriented Programming Using Java 11

3.1. Problems with Structured Program 11

3.2. Software Requirements 12

3.3. Hardware Requirements 12

4. System Design 13

4.1 UML diagrams 13

4.2 Use Case Diagram 13

4.3 Class Diagram 14

4.4 Collaboration Diagram 16

4.5 Sequence Diagram 17

5. Modules 18

5.1 Compressing a File 18

5.2 Decompressing a File 18

6. Implementation 19

6.1. Code 19

6.2. Screen Shots 23

7. Conclusion 26

7.1 Future Enhancement 26

8. Bibliography 27

**Chapter 1 – Introduction**

1. **INTRODUCTION:** The Domain “File Compression” lets you reduce the overall number of bits and bytes in a file so it can be transmitted faster over slower Internet connections, or take up less space on a disk. The software will be done using Core Java. The type of compression we will use here is called lossless compression. The user need not depend on third party software’s like WinZip, WinRAR, Stuff etc. the software can be used to compress files and they can be decompressed when the need arises. For implementing this Software, we want to use algorithms The main algorithm is: Huffman GZip algorithm.
2. **PROJECT OBJECTIVE:** The main objective of this File Compression project is to design a compression software which significantly reduces the size of a file so that it can be easily shared over mail even in slow internet speed. This software works in the same way as the WinRAR and WinZip, which are popular compression tools. Each byte of the file will be compressed and takes quite less memory on the disk. In this project, GZip algorithm will be used.
3. **EXISTING SYSTEM:** Existing system refers to the system that is being followed till now. The main disadvantage of this system is that the users depend on third party software’s like WinZip, WinRAR, Stuff etc. The existing system requires more computational time, more manual calculations, and the complexity involved in selection of features is high. The other disadvantages are lack of security of data, Deficiency of Data accuracy, Time consuming etc. Presently, the files cannot be sent or shared across mail or even across network if their size is above a certain limit. Even if the size lies in the allowable limit, then also sharing is difficult during low internet connection. These constraints create a lot of problem when something important needs to be shared urgently and the person is bounded by the size limit. Such situations require a compression tool which one should have handy.
4. **Draw backs of existing system:** 
   1. Lack of security of data.
   2. Deficiency of Data accuracy
   3. Time consuming.
   4. The users depend on third party software’s like WinZip, WinRAR, Stuff etc. To avoid all these limitations and make the working more accurately the system needs to be computerized.
5. **PROPOSED SYSTEM**: The aim of proposed system is to develop a system of improved facilities. The proposed system can overcome all the limitations of the existing system. The system provides data accuracy and save disc space. The existing system has several disadvantages and many more difficulties to work well. The proposed system tries to eliminate or reduce these difficulties up to some extent. The proposed system is file/folder compression or decompression based on the Huffman algorithm and GZip algorithm. The proposed system will help the user to consume time. The proposed system helps the user to work user friendly and he can easily do the file compression process without time lagging. The system is very simple in design and to implement. The system requires very low system resources and the system will work in almost all configurations. It has got following features Ensure data accuracy, minimize manual data entry, minimum time needed for the various processing, greater efficiency, better service. Advantages of Proposed System The system is very simple in design and to implement. The system requires very low system resources and the system will work in almost all configurations. It has got following features
   1. Ensure data accuracy and Save disk space
   2. Minimum time needed for the file compression
   3. Greater efficiency and Better Service
   4. Protection from virus and Easy to send via E-mail
   5. Maximum Compression rate is 2 GB.
   6. The user need not depend on third party software’s like WinZip, WinRAR, Stuff etc.

**CHAPTER 2 - LITERATURE SURVEY**

Many sources of information contain redundant data or data that adds little to the stored information. This results in tremendous amounts of data being transferred between client and server applications or computers in general. The obvious solution to the problems of data storage and information transfer is to install additional storage devices and expand existing communication facilities. One method to alleviate a portion of data storage and information transfer is through the representation of data by more efficient code.

This survey presents a brief introduction to data compression and decompression, and shows how to compress and decompress data, efficiently and conveniently, from within your Java applications using the java.util.zip package.

While it is possible to compress and decompress data using tools such as WinZip, gzip, and Java Archive (or jar), these tools are used as standalone applications. It is possible to invoke these tools from your Java applications, but this is not a straightforward approach and not an efficient solution. This is especially true if you wish to compress and decompress data on the fly (before transferring it to a remote machine for example).

Overview of Data Compression:

The simplest type of redundancy in a file is the repetition of characters. For example, consider the following string:

BBBBHHDDXXXXKKKKWWZZZZ

This string can be encoded more compactly by replacing each repeated string of characters by a single instance of the repeated character and a number that represents the number of times it is repeated. The earlier string can be encoded as follows:

4B2H2D4X4K2W4Z

Here "4B" means four B's, and 2H means two H's, and so on. Compressing a string in this way is called *run-length encoding*.

Another approach might be to store the image as a graphics metafile:

Rectangle 11, 3, 20, 5

This says, the rectangle starts at coordinate (11, 3) of width 20 and length 5 pixels.

The rectangular image can be compressed with run-length encoding by counting identical bits as follows:

|  |
| --- |
| 0, 40  0, 40  0,10 1,20 0,10  0,10 1,1 0,18 1,1 0,10  0,10 1,1 0,18 1,1 0,10  0,10 1,1 0,18 1,1 0,10  0,10 1,20 0,10  0,40 |

The first line above says that the first line of the bitmap consists of 40 0's. The third line says that the third line of the bitmap consists of 10 0's followed by 20 1's followed by 10 more 0's, and so on for the other lines.

Java provides the java.util.zip package for zip-compatible data compression. It provides classes that allow you to read, create, and modify ZIP and GZIP file formats. It also provides utility classes for computing checksums of arbitrary input streams that can be used to validate input data. This package provides one interface, fourteen classes, and two exception classes.

### Decompressing and Extracting Data from a ZIP file:

The java.util.zip package provides classes for data compression and decompression. Decompressing a ZIP file is a matter of reading data from an input stream. The java.util.zip package provides a ZipInputStream class for reading ZIP files. A ZipInputStream can be created just like any other input stream.

#### **ZIP File Properties**

The ZipEntry class describes a compressed file stored in a ZIP file. The various methods contained in this class can be used to set and get pieces of information about the entry. The ZipEntry class is used by the ZipFile and ZipInputStream to read ZIP files, and the ZipOutputStream to write ZIP files.

The Java ARchive (JAR) format is based on the standard ZIP file format with an optional manifest file. If you wish to create JAR files or extract files from a JAR file from within your Java applications, use the java.util.jar package, which provides classes for reading and writing JAR files. Using the classes provided by the java.util.jar package is very similar to using the classes provided by the java.util.zip package as described in this article. Therefore, you should be able to adapt much of the code in this article if you wish to use the java.util.jar package.

**Chapter 3 - TOOLS AND TECHNOLOGY**

**3.1 OBJECT ORIENTED PROGRAMMING AND JAVA**

Object-oriented Programming was developed because of limitations found in earlier approaches of programming. To appreciate what OOP does, we need to understand what these limitations are and how they arose from traditional programming.

**PROCEDURAL LANGUAGES:**

Pascal, C, Basic, FORTRAN, and similar languages are procedural languages. That is, each statement in the language tells the computer to do something: Get some Input, add these numbers,, divide by 6, display the output. A program in a procedural Language is a list of instructions. For very small programs no other organizing principle (often called a paradigm) is needed. The programmer creates the list of instructions, and the computer carries them out.

**DIVISION INTO FUNCTIONS:**

When programs become larger, a single list of instructions becomes unwieldy. Few programmers can comprehend a program of more than a few hundred statements unless it is broken down into smaller units. For this reason the function was adopted as way to make programs more comprehensible to their human creators. (The term functions are used in C++ and C. In other languages the same concept may be referred to as a subroutine, a subprogram, or a procedure.) A program is divided into functions, and (ideally, at least) each function has a clearly defined purpose and a clearly defined interface to the other functions in the program. The idea of breaking a program into functions can be further extended by grouping a number of functions together into a larger entity called a module, but the principle is similar:grouping a number of components that carry out specific tasks. Dividing a program into functions and modules is one of the cornerstones of structured programming, the somewhat loosely defined discipline that has influenced programming organization for more than a decade.

**3.2 Problems with Structured Programming:**

As programs grow ever larger and more complex, even the structured programming approach begins to show signs of strain. You may have heard about, or been involved in, horror stories of program development. The project is too complex, the schedule slips, more programmers are added, complexity increases, costs skyrocket, the schedule slips further, and disaster ensures. Analysing the reasons for these failures reveals that there are weaknesses in the procedural paradigm itself. No matter how well the structured programming approach is implemented, large programs become excessively complex.

**The object-oriented approach:**

The fundamental idea behind object-oriented languages is to combine into a single unit both data and the functions that operate on that data. Such a unit is called an object. An object’s functions, called member methods in Java, typically provide the only way to access its data. If you want to read the item and return the value to you, you call a member function in the object. It will read the item and return the value to you. You Can’t access the data directly. The data is hidden, so it is safe from accidental modification. Data and its functions are said to be encapsulated into a single entity. Data encapsulation and data hiding are key terms in the description of object oriented languages. If you want to modify the data in an object, you know exactly what functions interact with it: the member functions in the object. No other functions can access the data. This simplifies writing, debugging, and maintaining the program. A Java program typically consists of a number of objects, which communicate with each Other by calling one another’s members functions. We should mention that what are called member functions in C++ are called methods in Java. Also, data items are referred

to as instance variables. Calling an object’s member function is referred to as sending a

message to the object.

**3.3 SOFTWARE REQUIREMENTS:**

**Operating System:** Windows Operating System

**Languages:** Java 1.6 (latest version is recommended for better results/output)

**3.4 HARDWARE REQUIREMENTS:**

**Processor:** 600 MHz or above.

**RAM (SD/DDR):** 1 Gb

**Hard Disc:** 30GB

**Chapter 4 - SYSTEM DESIGN**

Systems design is the process of defining the architecture, components, modules, interfaces, and [data](http://en.wikipedia.org/wiki/Data) for a [system](http://en.wikipedia.org/wiki/System) to satisfy specified [requirements](http://en.wikipedia.org/wiki/Requirement). One could see it as the application of [systems theory](http://en.wikipedia.org/wiki/Systems_theory) to [product development](http://en.wikipedia.org/wiki/Product_development). [Object-oriented analysis and design](http://en.wikipedia.org/wiki/Object-oriented_analysis_and_design) methods are becoming the most widely used methods for computer systems design.

**UML Diagrams**

UML stands for Unified Modelling Language which is used in object oriented software engineering. Although typically used in software engineering it is a rich language that can be used to model an application structures, behaviour and even business processes. There are **14 UML diagram types** to help you model these behaviors.

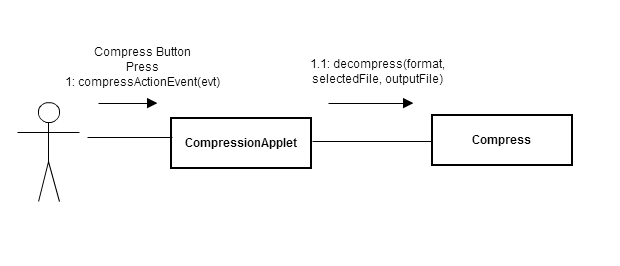
**Use Case Diagrams**

As the most known diagram type of the behavioural UML diagrams, Use case diagrams give a graphic overview of the actors involved in a system, different functions needed by those actors and how these different functions are interacted.

It’s a great starting point for any project discussion, because you can easily identify the main actors involved and the main processes of the system

UML Diagrams for our application are as follows:

**Use Case Diagram:**

****

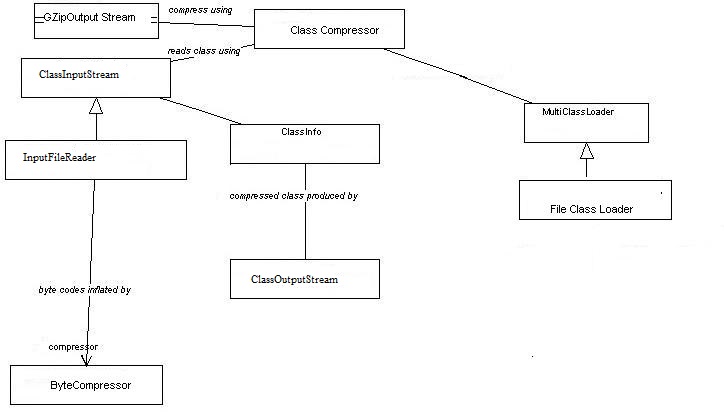
**Class Diagram:**

Class diagrams are arguably the most used UML diagram type. It is the main building block of any object-oriented solution. It shows the classes in a system, attributes and operations of each class and the relationship between each class.

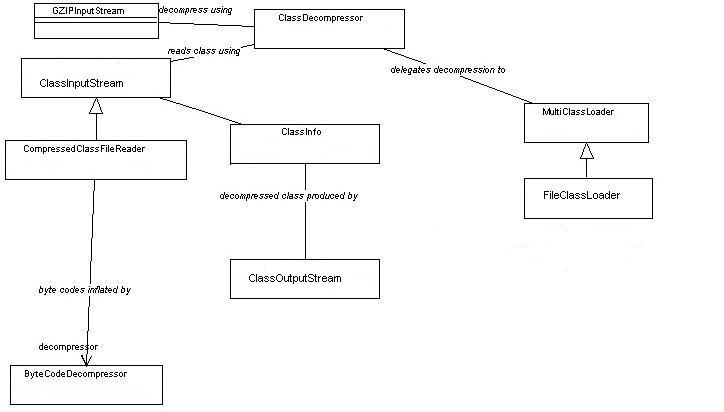
In most modelling tools, a class has three parts, name at the top, attributes in the middle and operations or methods at the bottom. In large systems with many related classes, classes are grouped together to create class diagrams. Different relationships between classes are shown by different types of arrows.

Below is an image of a class diagram for compression and decompression part of the project.

**Class Diagram for Compression Module**

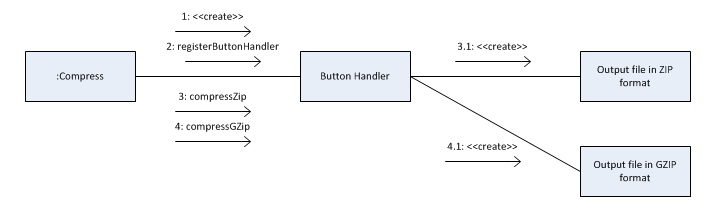


**Class Diagram for Decompression Module**



**Collaboration Diagram:**

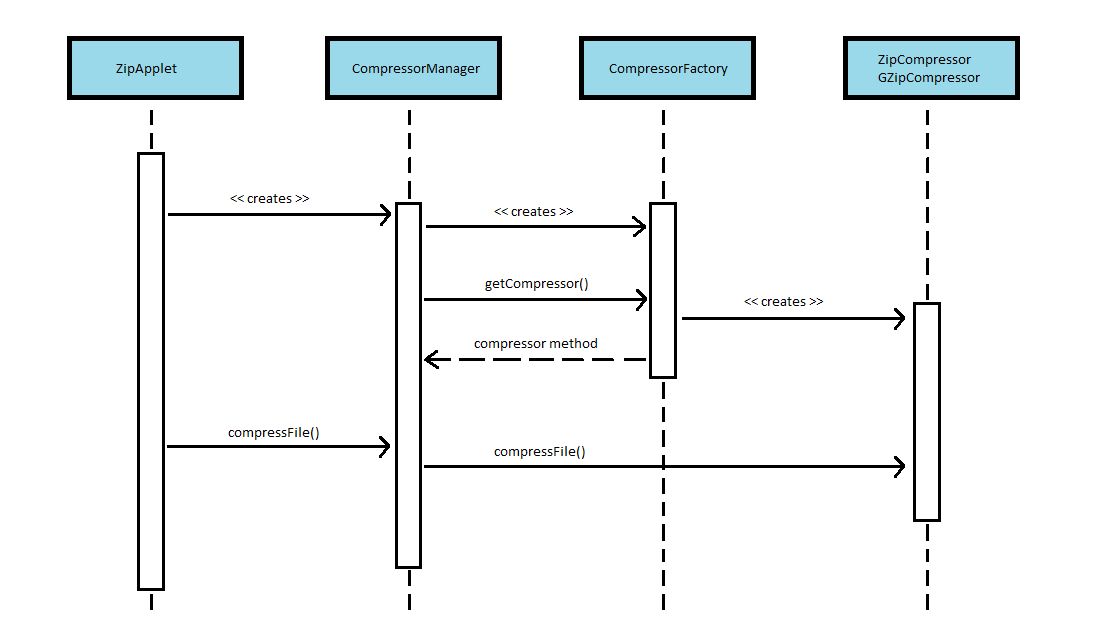
collaboration diagram in UML are is similar to sequence diagrams, but the focus is on messages passed between objects. The same information can be represented using a sequence diagram and different objects.

****

**`**

**Sequence Diagram:**

Sequence diagrams in UML show how objects interact with each other and the order those interactions occur. It’s important to note that they show the interactions for a particular scenario. The processes are represented vertically and interactions are show as arrows.

****

**CHAPTER 5 – MODULES**

The Domain File Compression mainly include 3 modules

• Compress A File or Folder

• De-Compress the file or folder

• View files in the compressed file

6.1 Compress file or folder: This module helps us to compress a file or folder. The compressed file will have a extension that has been given at the development time. We can send the compressed file over the internet so that users having this software can decompress it.

6.2 Decompress a file or folder: This is the reverse process of file compression. Here we can decompress the compressed file and get the original file.

6.3 View files in the compressed file Here we can view the list of files inside our compressed file. We can view the files before decompressing and decide to decompress or not.

**CHAPTER 6 – IMPLEMENTATION**

The implementation stage of any project is a true display of the defining moments that make a project a success or a failure. The implementation stage is defined as the system or system modifications being installed and made operational in a production environment. The phase is initiated after the system has been tested and accepted by the user. This phase continues until the system is operating in production in accordance with the defined user requirements.

**Code**

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.zip.\*;

import java.util.zip.GZIPInputStream;

import java.util.zip.GZIPOutputStream;

import java.awt.\*;

import java.awt.event.\*;

class c extends Frame implements ActionListener,WindowListener

{

FileInputStream fis;

FileOutputStream fos;

GZIPOutputStream df;

GZIPInputStream ff;

byte[] buffer;

int len;

Label l,l2,l3;

Button b,b2;

TextField t,t2;

c()

{

setLayout(null);

l2=new Label("Source path");

l=new Label("File Compression and Decompression");

l3=new Label("Destinaton");

b=new Button("COMPRESS");

b2=new Button("DECOMPRESS");

t=new TextField(100);

t2=new TextField(100);

setVisible(true);

setSize(500,500);

setBackground(Color.ORANGE);

addWindowListener(this);

b.addActionListener(this);

b2.addActionListener(this);

l.setBounds(60,20,350,50);

l2.setBounds(25,75,75,50);

l3.setBounds(25,150,75,50);

t.setBounds(110,75,200,30);

t2.setBounds(110,150,200,30);

b.setBounds(110,225,100,25);

b2.setBounds(255,225,100,25);

add(l);

add(b);

add(t);

add(t2);

add(l2);

add(l3);

add(b2);

}

public void windowClosed(WindowEvent ev){}

public void windowOpened(WindowEvent ev){}

public void windowActivated(WindowEvent ev){}

public void windowDeactivated(WindowEvent ev){}

public void windowIconified(WindowEvent ev){}

public void windowDeiconified(WindowEvent ev){}

public void windowClosing(WindowEvent ev)

{

System.exit(0);

}

public void actionPerformed(ActionEvent e1)

{

if(e1.getSource()==b)

{

try {

fis = new FileInputStream(t.getText());

fos = new FileOutputStream(t2.getText());

df = new GZIPOutputStream(fos);

buffer = new byte[2048];

int len;

while((len=fis.read(buffer)) != -1){

df.write(buffer, 0, len);

}

//close resources

df.close();

fos.close();

fis.close();

} catch (Exception ee)

{

t.setText("file not found enter valid location");

}

t.setText("File Successfully Compressed");

}

else if(e1.getSource()==b2)

{

try {

fis = new FileInputStream(t.getText());

ff = new GZIPInputStream(fis);

fos = new FileOutputStream(t2.getText())

buffer = new byte[2048];

int len;

while((len=ff.read(buffer)) != -1){

fos.write(buffer, 0, len);

}

//close resources

ff.close();

fos.close();

fis.close();

} catch (Exception eg)

{

t.setText("file not found enter valid location");

}

t.setText("File Successfully deCompressed");

}

}

public static void main(String args[])

{

c a1 = new c();

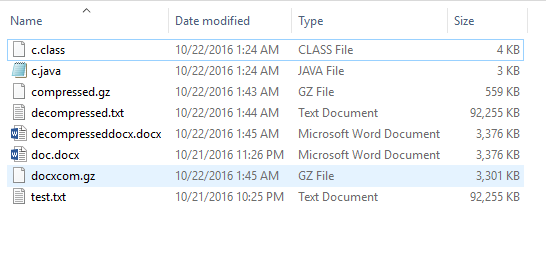
}

}

**CHAPTER 7 - Testing**

**Output Screens:**

**Test Case Output Screen:**

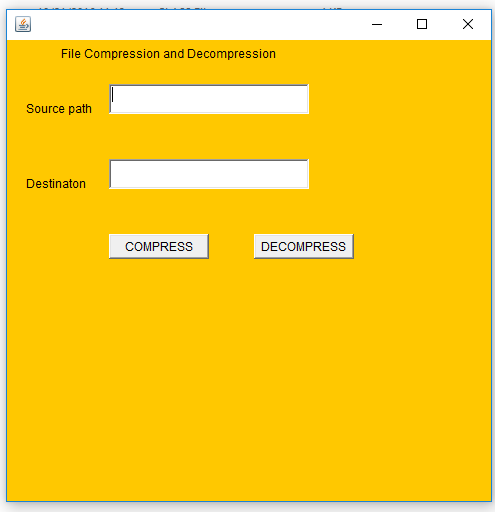
****

The above test output screen shows a sample text document that has been compressed and decompressed using the code developed during the evolution of our project.

The .gz file shows that the sample text document has been successfully compressed to a size of 3301KB and 559KB from the original size of 3376KB and 92255KB for the document word file and text file respectively

**AWT GUI:**

The following AWT GUI screen for our code is developed for easy way for compression decompression without the use of command prompt.

****

**CHAPTER 8 – CONCLUSION**

The final AWT GUI developed using the JAVA Zip packages and GZip algorithm give the user a better way of compressing files so that they can save a good amount of memory space with an added advantage of text and document file encryption.

Future Enhancements**:**

* The compression and decompression can be written in a native language such as 'C', to achieve better performance.
* By change the ByteCode compression algorithm to generate patterns dynamically for better compression.
* The AWT GUI thus developed can be fully developed into a Mobile Application so that users can instantly compress/decompress their files on the go.

**BIBLIOGRAPHY**

**References:**

1. **“Java: The Complete Reference”** by Herbert Schildt.
2. Md. RubaiyatHasan, “**Data Compression using Huffman based LZW Encoding Technique**”
3. S. Kaur and V.S.Verma,”**Design and Implementation of LZW Data Compression Algorithm.”**
4. <http://www.oracle.com/technetwork/articles/java/compress-1565076.html>.