AHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH

**Table of Contents**

**Definition of the Problem.**

## Description of the scope of the problem to be tackled

For my project I decided to make a program that compiles and decompiles strings of text. I chose it as it would be a good learning experience for me as it would help me understand how encoding works while also giving me a chance to learn about java’s GUI library. I used the Java programming language as it would help me learn more about Object Oriented Programming, it would also allow the project to be built on multiple different platforms due to how java is written. Having my project computerized allows for quick and easy conversion, without it I would have to learn many different types of ways to convert words into a chosen converter.

## Statement of the results required

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Type | Found in | Description |
| choices | String Array | main.java | Stores what encoders the users can use, these show up in the encoder/decoder dropdown. |
| selectedchoice | String | main.java | Gets the current choice that was selected in the encoder/decoder dropdown. |
| choice | int | FileChooser.java | Changes depending on which button the user presses in the file chooser. |
| output | String | Encoder.java  Decoder.java | When a user uses an encoder or decoder the outputted value will be put into the output variable. |
| getChar | var | Encoder.java  Decoder.java |  |
|  |  |  |  |
|  |  |  |  |

Chapter 2: Solution of the Problem

## Algorithm Design

1. The **main** class is used as the body of the project, this is where the main window and user interface is drawn to the screen, it is also where the user will be able to use the project.
2. The **info** class draws a window to the screen and shows information about the project.
3. The **Encoder** class handles all the logic for the five encoders that I chose to program for this project.
4. The **Decoder** class handles all the logic for the five decoders, it does the opposite of the **Encoder** class.
5. The **FileChooser** class handles all of the logic regarding the file chooser that allows the user to open a file and save a file.

* The **main** class has six functions which include:

1. main(): This function is run when the program is started and creates a JFrame and a JPanel so that UI elements can be added later on.
2. drawUI(): This function is triggered after the window and the UI panel are created, this function adds all of the UI elements to the screen and sets their positions accordingly.
3. encodeText(String text): This function is triggered if the user presses the **Encode** button. This function checks what encoder option the user has selected and the inputted string and encodes it using the selected encoder.
4. decodeText(String text): This function is similar to the last one except it is triggered when the user presses the **Decode** button and instead of encoding the string, it decodes it with the selected decoder.
5. copytoclipboard(String text): This function is triggered when the user presses the **Copy to Clipboard** button. This function copies the outputted string into the user’s clipboard.
6. setOutput(String output): This function simply sets the output field to the desired string. This function is mainly used in external classes to allow for an easier and more readable way to change the output field’s contents.

* The **Info** class only has one function:

1. infoMenu(): This function opens the info window and creates all of its UI elements.

* The **Encoder** class has six functions which include:

1. outputInfo(String input, String output, String encoder): When triggered, this function will output information about the inputted string, the outputted string and the encoder.
2. Base64(String text): When triggered this function will use the Base64 class found in at **java.util.Base64** to encode the given string. It will then output the encoded string into the console.
3. DAscii(String text): When triggered this function will convert the given string into a Decimal ASCII string.
4. Binary(String text): When triggered, this function will convert the inputted string to its binary equivalent. It will then output the binary value.
5. CaesarCipher(String text): When triggered this function will convert the given string into its Caesar Cipher equivalent, it does this by increasing the value of every character in the string which will give the next character. It will then output the encoded string.
6. Reverse(String text): When triggered this function will reverse the given string. It will then output the result.

* The **Decoder** class has six functions which include:

1. outputInfo(String input, String output, String encoder): When triggered, this function will output information about the inputted string, the outputted string and the decoder.
2. Base64(String text): When triggered this function will use the Base64 class found in at **java.util.Base64** to decode the given string. It will then output the decoded string into the console.
3. DAscii(String text): When triggered this function will convert a Decimal ASCII string into a readable string.
4. Binary(String text): When triggered, this function will convert the binary input given into readable text.It will then output the text.
5. CaesarCipher(String text): When triggered this function will convert the Caesar Cipher input given into a readable string, it does this by decreasing the value of every character in the string which will give the last character. It will then output the decoded string.
6. Reverse(String text): When triggered this function will reverse the given string. It will then output the result.

* The **FileChooser** class has two functions which include:

1. openFileChooser(): This function will open a new window with a file chooser that only allows the user to open files, this is useful when the user wants to get the text from a file and put it into the input field.
2. saveFileChooser(): This function will open a new window with a file chooser that only allows the user to save files, this is useful when the user wants to save the outputted value/s in the output field to a text file.

ADD FLOW CHART HERE

## Computer listing of the program

## Details of any special design features

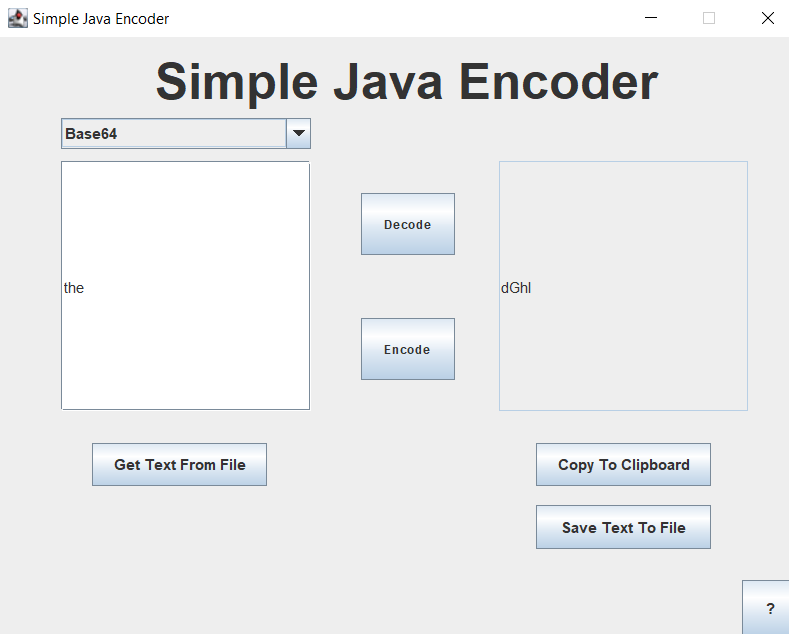
* For my project I decided to create a user interface to control everything, this allows for easier control over the program as it is much easier to use a user interface rather than using a terminal window to control the project.
* The main things that allow the program to work are the **Encoder** and **Decoder** classes**,** these hold functions that encode and decode inputted values.

Chapter 3: Running the Program

A screenshot of a computer

Description automatically generated

The above image shows the main screen of the program.



The above image shows the user using the Base64 encoder, this is done by inputting a string into the left field and pressing the Encode button.

ADD MORE SCREENSHOTS

## Plan of Test Data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Plan** | **Test Data** | **Expected Output** | **Actual Output** | **Remarks** |
|  |  |  |  |  |

Chapter 4: User Instruction

## Loading and using the program

Chapter 5: Comments and Conclusions

## Limitations and Improvement

From making this project I have learnt how the Java programming language works. I have also learnt how to create a user interface with the use of Java Swing, the inner workings of encoders and decoders and how to create and manage file choosers on Java Swing.

I enjoyed making this program because it was a new learning experience for me since I hadn’t used Java before working on this project, it was fun to find solutions and workarounds for any problems I had and finding out the basics of how encoding and decoding of values work.

I feel like I was able to meet my expectations for this project as I wanted to create a simple yet interesting project since I knew that with my level of knowledge of Java it would be difficult for me to create something more complex.

I think that one of the main improvements I could have made to my program was improving the code and giving the program a better user interface with the use of icons and different colours.