**Simple Java Encoder**



Name: Jurgen Arnaud

Class: 11 Arnold

Subject: Computing Coursework 2023-2024

ID:

**Table of Contents**

**Chapter 1: Definition of the Problem 3**

Description of the scope of the problem to be tackled 3

Statement of the results required 3

**Chapter 2: Solution of the Problem 5**

Algorithm Design 5

Computer listing of the program 14

Details of any special design features 34

**Chapter 3: Running the Program 35**

Evidence that the solution works 35

Plan of Test Data 37

**Chapter 4: User Instruction 38**

Loading the Program 38

**Chapter 5: Comments and Conclusions 42**

Limitations and Improvement 42

**Chapter 1: 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

This program starts up with a main screen where different buttons and text fields are displayed. When a user inputs a string into the left text field, selects an converter and presses encode or decode the text will go through the selected converter and display the final result in the input field located on the right side.

If the user wishes to copy the converted text they may do so by pressing on the “Copy to Clipboard” button located just under the right text field.

The user can load text files into the left field by pressing the “Get Text From File” button found underneath the left field. The user can also save the converted text into the right field by pressing the “Save Text To File” button located underneath the copy to clipboard button.

All of the previously mentioned actions have different variables that help them produce the expected results. Those variables can be found in the table below along with what type of variable they are, where they are found and a description of what they do.

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | Gets a character from the inputted string. |

**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.

On the next page you will find the flowchart for the project. The flowchart is a plan for the project and outlines all of its features and how the project works behind the scenes while also being simple and very readable due to the fact that it uses shapes and arrows.

A black and white background with white squares

Description automatically generated with medium confidence

A black background with white rectangles

Description automatically generatedA screenshot of a computer screen

Description automatically generated

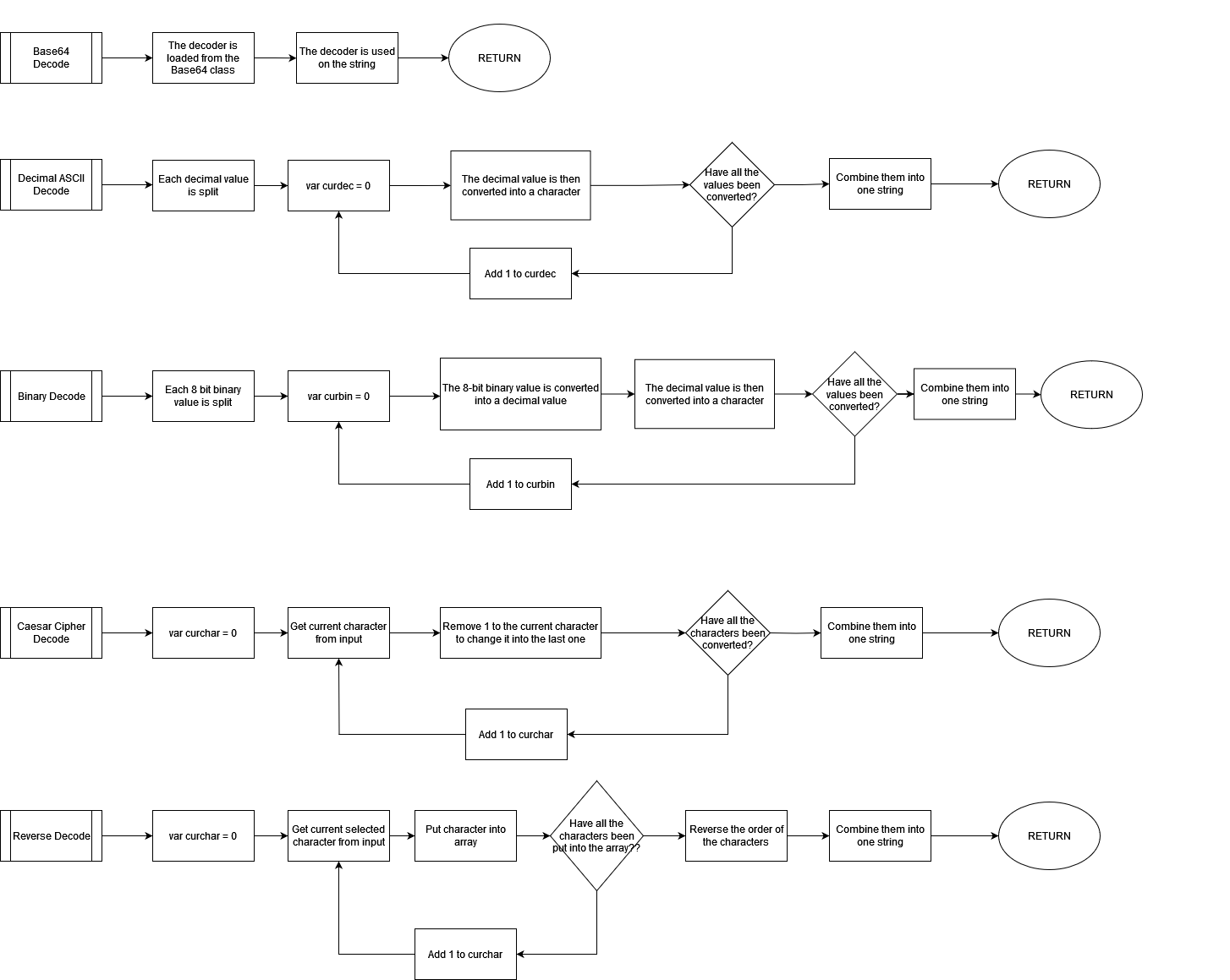
A black background with white rectangles

Description automatically generatedA group of white rectangular shapes

Description automatically generated

A group of white rectangles

Description automatically generated



## Computer listing of the program

**main.java**

**import javax.swing.\*;**

**import java.awt.\*;**

**import java.awt.event.\*;**

**import java.awt.datatransfer.StringSelection;**

**import java.awt.datatransfer.Clipboard;**

**public class main {**

**static JFrame window;**

**static JPanel panel;**

**static JLabel title;**

**static JButton openfile;**

**static JButton savefile;**

**static JButton encodebtn;**

**static JButton decodebtn;**

**static JButton copybtn;**

**static JTextField inputfield;**

**static JTextField outputfield;**

**static JComboBox<String> encoderdropdown;**

**static String[] choices = {"Base64", "Decimal ASCII", "Binary", "Caesar Cipher", "Reverse"};**

**static JButton infobutton;**

**public static void main(String[] args) throws Exception {**

**// Creates a window.**

**window = new JFrame("Simple Java Encoder");**

**window.setResizable(false);**

**// Creates the UI panel so that the interface can be added to the window.**

**panel = new JPanel();**

**panel.setPreferredSize(new Dimension(640, 480));**

**panel.setLayout(null);**

**drawUI();**

**window.add(panel);**

**window.pack();**

**window.setVisible(true);**

**}**

**// A function that draws the UI to the screen.**

**static void drawUI()**

**{**

**title = new JLabel("Simple Java Encoder");**

**title.setFont(new Font(title.getFont().getName(), Font.BOLD, 40));**

**title.setBounds(125, 10, 1000, 50);**

**// A dropdown element that allows the user to select an encoder.**

**encoderdropdown = new JComboBox<String>(choices);**

**encoderdropdown.setBounds(50, 65,200, 25);**

**// The input field that allows the user to input a string.**

**inputfield = new JTextField();**

**inputfield.setBounds(50, 100, 200, 200);**

**// The output field that shows the user the converted input.**

**outputfield = new JTextField();**

**outputfield.setBounds(400, 100, 200, 200);**

**outputfield.setEditable(false);**

**// The encode button, when pressed it will encode the string in the input field and output it in the output field.**

**encodebtn = new JButton("Encode");**

**encodebtn.setPreferredSize(new Dimension(100, 100));**

**encodebtn.setBounds(290, 225, 75, 50);**

**encodebtn.setFont(new Font(encodebtn.getFont().getName(), Font.BOLD, 10));**

**encodebtn.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**encodeText(inputfield.getText().trim());**

**}**

**});**

**// The decode button, when pressed it will decode the string in the input field and output it in the output field.**

**decodebtn = new JButton("Decode");**

**decodebtn.setPreferredSize(new Dimension(100, 100));**

**decodebtn.setBounds(290, 125, 75, 50);**

**decodebtn.setFont(new Font(decodebtn.getFont().getName(), Font.BOLD, 10));**

**decodebtn.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**decodeText(inputfield.getText());**

**}**

**});**

**// The Copy to Clipboard button, when pressed it will copy the string in the output field to the clipboard.**

**copybtn = new JButton("Copy To Clipboard");**

**copybtn.setPreferredSize(new Dimension(100, 100));**

**copybtn.setBounds(500 - 70, 325, 140, 35);**

**copybtn.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**copytoclipboard(outputfield.getText());**

**}**

**});**

**// The Info button, when pressed it will display a new window showing information about the project.**

**infobutton = new JButton("?");**

**infobutton.setBounds(640 - 45, 480 - 45, 45, 45);**

**infobutton.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**Info.infoMenu();**

**}**

**});**

**// The Get Text From File button, when pressed it will open a file chooser window that allows the user to choose a file to open into the input field.**

**openfile = new JButton("Get Text From File");**

**openfile.setPreferredSize(new Dimension(100, 100));**

**openfile.setBounds(145 - 70, 325, 140, 35);**

**openfile.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**FileChooser.openFileChooser();**

**}**

**});**

**// The Save Text To File button, when pressed it will open a file chooser window that allows the user to save the contents of the output folder to a file.**

**savefile = new JButton("Save Text To File");**

**savefile.setPreferredSize(new Dimension(100, 100));**

**savefile.setBounds(500 - 70, 375, 140, 35);**

**savefile.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**FileChooser.saveFileChooser();**

**}**

**});**

**// Adding UI elements to the window.**

**panel.add(title);**

**panel.add(openfile);**

**panel.add(savefile);**

**panel.add(inputfield);**

**panel.add(outputfield);**

**panel.add(encodebtn);**

**panel.add(decodebtn);**

**panel.add(copybtn);**

**panel.add(encoderdropdown);**

**panel.add(infobutton);**

**}**

**// A function that loads the text and the encoder that was selected by the user.**

**static void encodeText(String text)**

**{**

**outputfield.setText("");**

**String selectedchoice = String.valueOf(encoderdropdown.getSelectedItem());**

**switch (selectedchoice)**

**{**

**case "Base64":**

**Encoder.Base64(text);**

**break;**

**case "Decimal ASCII":**

**Encoder.DAscii(text);**

**break;**

**case "Binary":**

**Encoder.Binary(text);**

**break;**

**case "Caesar Cipher":**

**Encoder.CaesarCipher(text);**

**break;**

**case "Reverse":**

**Encoder.Reverse(text);**

**break;**

**}**

**}**

**// A function that loads the text and the decoder that was selected by the user.**

**static void decodeText(String text)**

**{**

**outputfield.setText("");**

**String selectedchoice = String.valueOf(encoderdropdown.getSelectedItem());**

**switch (selectedchoice)**

**{**

**case "Base64":**

**Decoder.Base64(text);**

**break;**

**case "Decimal ASCII":**

**Decoder.DAscii(text);**

**break;**

**case "Binary":**

**Decoder.Binary(text);**

**break;**

**case "Caesar Cipher":**

**Decoder.CaesarCipher(text);**

**break;**

**case "Reverse":**

**Decoder.Reverse(text);**

**break;**

**}**

**}**

**// A function that copies a string to the user's clipboard.**

**static void copytoclipboard(String text)**

**{**

**StringSelection stringSelection = new StringSelection(text);**

**Clipboard clipboard = Toolkit.getDefaultToolkit().getSystemClipboard();**

**clipboard.setContents(stringSelection, null);**

**}**

**// A function that sets the output of the output field to the inputted string. This is mainly used in other classes to easily change the output field.**

**public static void setOutput(String output)**

**{**

**outputfield.setText(output);**

**}**

**}**

**Info.java**

**import javax.swing.JFrame;**

**import javax.swing.JLabel;**

**import javax.swing.JPanel;**

**import java.awt.\*;**

**public class Info {**

**static JFrame window;**

**static JPanel panel;**

**static JLabel title;**

**static JLabel label;**

**public static void infoMenu()**

**{**

**// Makes a window and adds a UI panel.**

**window = new JFrame("Information");**

**window.setResizable(false);**

**panel = new JPanel();**

**panel.setPreferredSize(new Dimension(330, 240));**

**panel.setLayout(null);**

**// Makes text elements.**

**title = new JLabel("Information");**

**title.setFont(new Font(title.getFont().getName(), Font.BOLD, 20));**

**title.setBounds(320 / 2 - 50, 10, 200, 50);**

**label = new JLabel("<html>Project made for the 2022-2024 Computing<br/>Course. Enter an input in the left field, press the<br/>ENCODE button to encode your input or press<br/>DECODE to decode your input.<html>");**

**label.setFont(new Font(title.getFont().getName(), Font.PLAIN, 15));**

**label.setBounds(10, 0, 1000, 200);**

**// Adds UI elements to the panel.**

**window.add(panel);**

**panel.add(title);**

**panel.add(label);**

**window.pack();**

**window.setVisible(true);**

**}**

**}**

**FileChooser.java**

**import java.io.IOException;**

**import java.nio.file.Files;**

**import java.nio.file.Path;**

**import java.nio.file.StandardOpenOption;**

**import java.io.BufferedWriter;**

**import java.io.File;**

**import javax.swing.JFileChooser;**

**public class FileChooser {**

**// Function that loads and opens the file chooser that allows the user to open files.**

**public static void openFileChooser()**

**{**

**JFileChooser filechooser = new JFileChooser();**

**int choice = filechooser.showOpenDialog(filechooser);**

**filechooser.cancelSelection();**

**System.out.println("File Chooser Opened");**

**if (choice == JFileChooser.OPEN\_DIALOG)**

**{**

**System.out.println("Opening File...");**

**System.out.println(filechooser.getSelectedFile());**

**try**

**{**

**// Reads the file and adds the contents of the file to the input field.**

**main.inputfield.setText(Files.readString(filechooser.getSelectedFile().toPath()));**

**}**

**catch (IOException e)**

**{}**

**}**

**}**

**// Function that loads and opens the file chooser that allows the user to save files.**

**public static void saveFileChooser()**

**{**

**JFileChooser filechooser = new JFileChooser();**

**int choice = filechooser.showSaveDialog(filechooser);**

**//filechooser.setFileFilter(filechooserfilter);**

**// filechooser.cancelSelection();**

**if (choice == JFileChooser.APPROVE\_OPTION)**

**{**

**System.out.println("Saving File...");**

**try**

**{**

**//Makes a file or overrides a file with the given name in the filechooser**

**Path path = filechooser.getSelectedFile().toPath();**

**File file = new File(path.toString() + ".txt");**

**file.createNewFile();**

**System.out.println("MADE NEW FILE: " + file.getName());**

**System.out.println(filechooser.getCurrentDirectory().toString());**

**// Writes to the file**

**BufferedWriter writer = null;**

**writer = Files.newBufferedWriter(file.toPath(), StandardOpenOption.APPEND);**

**writer.write(main.outputfield.getText());**

**writer.newLine();**

**writer.close();**

**}**

**catch (IOException e)**

**{**

**System.out.println("An error occurred.");**

**e.printStackTrace();**

**}**

**}**

**}**

**}**

**Encoder.java**

**import java.util.Base64;**

**public class Encoder {**

**static String output = "";**

**// Outputting information about the encoder to the console.**

**static void outputInfo(String input, String output, String encoder){**

**System.out.println("String: " + input + " ---" + encoder + "---> Output: " + output + "\n");**

**}**

**// Conveting a string to Base64**

**public static void Base64(String text){**

**output = Base64.getEncoder().encodeToString(text.getBytes());**

**main.setOutput(output);**

**outputInfo(text, output, "Base64");**

**}**

**// Converting a string to Decimal ASCII**

**public static void DAscii(String text){**

**int[] asciiarray = new int[text.length()];**

**for (int i = 0; i < text.length(); i++) {**

**var getChar = (text.charAt(i));**

**asciiarray[i] = getChar;**

**output = (main.outputfield.getText() + " " + asciiarray[i]).trim();**

**main.setOutput(output);**

**}**

**outputInfo(text, main.outputfield.getText(), "Decimal ASCII");**

**}**

**// Converting a string to Binary ASCII**

**public static void Binary(String text)**

**{**

**int[] asciiarray = new int[text.length()];**

**for (int i = 0; i < text.length(); i++) {**

**var getChar = (text.charAt(i));**

**asciiarray[i] = getChar;**

**output = (main.outputfield.getText() + Integer.toBinaryString(text.charAt(i)) + " ");**

**main.setOutput(output);**

**}**

**outputInfo(text, main.outputfield.getText(), "Binary");**

**}**

**// Encodes a string using the Caesar Cipher**

**public static void CaesarCipher(String text)**

**{**

**output = "";**

**for (int i = 0; i < text.length(); i++) {**

**char getChar = (text.charAt(i));**

**getChar += 1;**

**if (getChar != 33)**

**{**

**output = String.valueOf(getChar);**

**}**

**else**

**{**

**output = " ";**

**}**

**main.setOutput(main.outputfield.getText() + "" + output);**

**}**

**outputInfo(text, main.outputfield.getText(), "Caesar Chipher");**

**}**

**// Reverses the string and outputs the value.**

**public static void Reverse(String text)**

**{**

**output = "";**

**for (int i = 0; i < text.length(); i++)**

**{**

**char getChar = text.charAt(i);**

**output = getChar + output;**

**}**

**main.setOutput(main.outputfield.getText() + "" + output);**

**outputInfo(text, main.outputfield.getText(), "Reverse");**

**}**

**}**

**Decoder.java**

**import java.util.Base64;**

**public class Decoder {**

**static String output = "";**

**// Outputting information about the decoder to the console.**

**static void outputInfo(String input, String output, String encoder){**

**System.out.println("String: " + input + "\n" + "Output: " + output + "\n" + "Encoder: " + encoder);**

**}**

**// Conveting Base64 to a string**

**public static void Base64(String text)**

**{**

**output = new String(Base64.getDecoder().decode(text));**

**main.setOutput(output);**

**outputInfo(text, output, "Base64");**

**}**

**// Converting Decimal ASCII to a string**

**public static void DAscii(String text)**

**{**

**String[] asciiarray = text.trim().split(" ");**

**for (int i = 0; i < asciiarray.length; i++)**

**{**

**char dectostring = (char)Integer.parseInt(asciiarray[i]);**

**System.out.println(dectostring);**

**output = main.outputfield.getText() + dectostring;**

**output.trim();**

**main.setOutput(output);**

**}**

**}**

**// Converting Binary ASCII to a string**

**public static void Binary(String text)**

**{**

**String[] binaryarray = text.trim().split(" ");**

**for (int i = 0; i < binaryarray.length; i++)**

**{**

**int bintodec = Integer.valueOf(binaryarray[i], 2);**

**char dectostring = (char)bintodec;**

**System.out.println(dectostring);**

**output = main.outputfield.getText() + dectostring;**

**output.trim();**

**main.setOutput(output);**

**}**

**}**

**// Decodes a string using the Caesar Cipher**

**public static void CaesarCipher(String text)**

**{**

**output = "";**

**for (int i = 0; i < text.length(); i++) {**

**char getChar = (text.charAt(i));**

**getChar -= 1;**

**if (getChar != 31)**

**{**

**output = String.valueOf(getChar);**

**}**

**else**

**{**

**output = " ";**

**}**

**main.setOutput(main.outputfield.getText() + "" + output);**

**}**

**outputInfo(text, main.outputfield.getText(), "Caesar Chipher");**

**}**

**// Reverses the string and outputs the value.**

**public static void Reverse(String text)**

**{**

**var output = "";**

**for (int i = 0; i < text.length(); i++)**

**{**

**char getChar = text.charAt(i);**

**output = getChar + output;**

**}**

**main.setOutput(main.outputfield.getText() + "" + output);**

**outputInfo(text, main.outputfield.getText(), "Reverse");**

**}**

**}**

## Details of any special design features

* For my project I decided to create a simple 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 which allows it to be more accessible to more people.
* The main things that allow the program to work are the **Encoder** and **Decoder** classes**,** these hold functions that encode and decode inputted values.
* Another special design featured in my project is a file chooser that allows the user to save the outputted value into a text file and also allows the user to load a text file into the left text field.
* My project utilizes action listeners and external classes which allow the project to function properly. I have also implemented different classes for different actions to make the code much easier to understand.

**Chapter 3: Running the Program**

## Evidence that the solution works

A screenshot of a computer

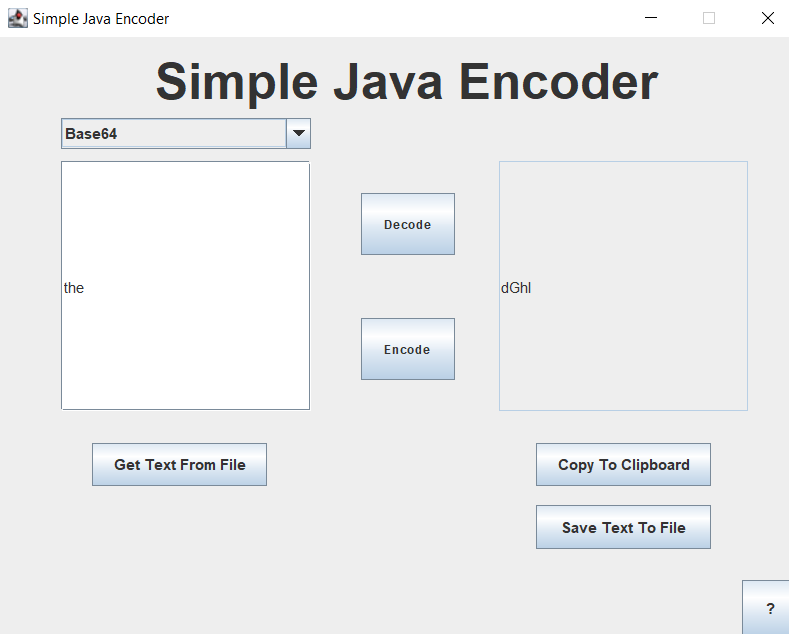
Description automatically generated

This image shows the main screen of the program

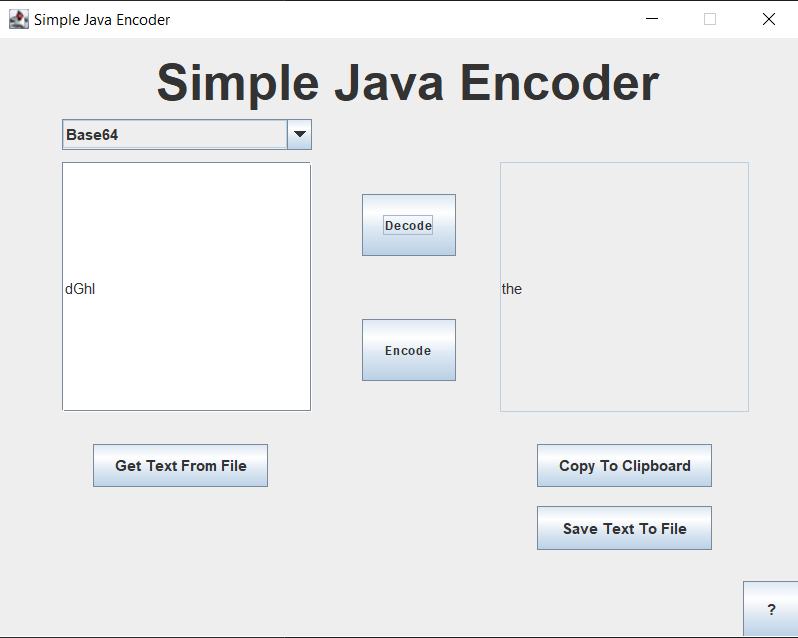
A screenshot of a computer

Description automatically generated

This image shows the user selecting the Base64 encoder using the dropdown menu



This image shows the Base64 encoder working



This image shows the Base64 decoder working

## Plan of Test Data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Plan** | **Test Data** | **Expected Output** | **Actual Output** | **Remarks** |
| Information button works on the main menu | Pressing the information button | A window opens with information about the project | Window opens and shows information | Test Passed |
| Putting a character that is not usually on a keyboard | Putting the “Ø” symbol through the Caesar Cipher encoder | The ” Ù” symbol is outputted into the right text field | The ”Ù” was outputted | Test Passed |
| Checking if the save file chooser opens | Pressing the “Save To Text File” button | A file chooser window opens that allows the user to save a file to the desired location. | File chooser opened and allowed the user to save a file. | Test Passed |
| Checking if the load file chooser opens | Pressing the “Get Text From File” button | A file chooser window opens that allows the user to load a text file into the project | File chooser opened and allowed the user to load a file. | Test Passed |
| Copying the outputted text to the clipboard. | Pressing the “Copy To Clipboard” button | The outputted text is copied into the user’s clipboard. | Outputted text was copied into the clipboard | Test Passed |

**Chapter 4: User Instruction**

## Loading and using the program

**If you do not have BlueJ currently installed please follow the instructions below:**

1. To install the program, you must first install BlueJ from [**https://www.bluej.org/**](https://www.bluej.org/) for your operating system.

A screenshot of a computer

Description automatically generated

1. Open the setup wizard and go through the installation process.

A screenshot of a computer

Description automatically generated

**Running the program**

1. Open the folder containing the **java files** and the **package.bluej** and double click the **package.bluej** file to open the project

A screenshot of a computer

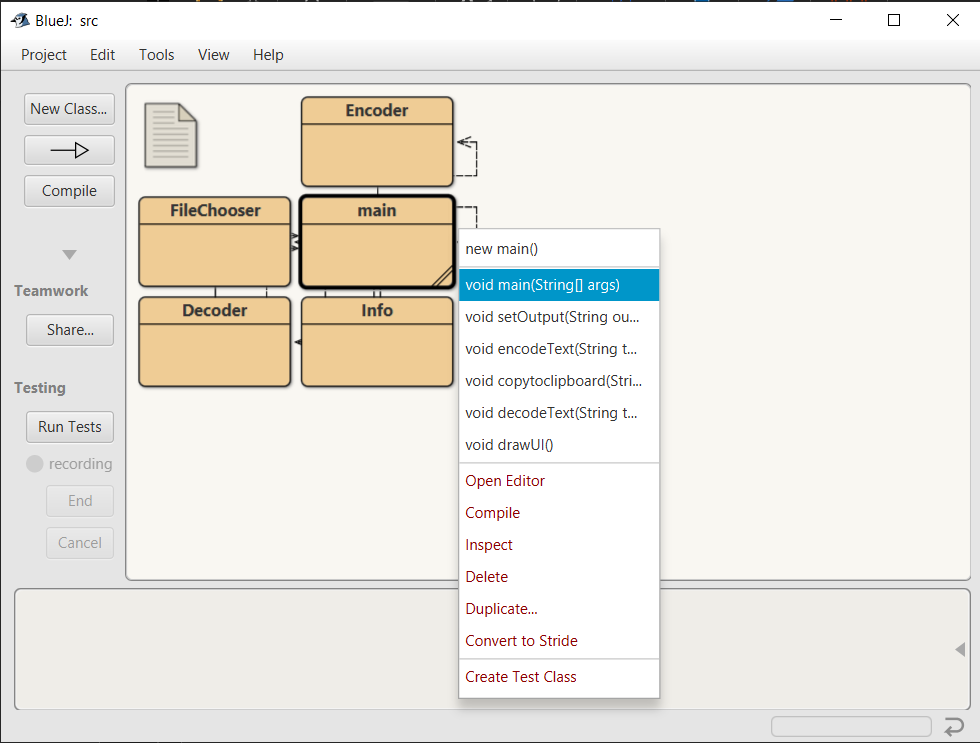
Description automatically generated

1. Now that the project is opened, locate the top toolbar and press the **Tools** dropdown menu, then press on the compile button, this will compile all of the class files so that you will be able to run the program.

A screenshot of a computer

Description automatically generated

1. Right click the **main** button and press the **void main(String args[])** button.



1. This window should now be opened on your screen, simply press the **OK** button to run the program.

A screenshot of a computer

Description automatically generated

1. Congratulations! You should now be running the program, for more information on how to use the program please press the information button located at the bottom right of the screen or refer to **Chapter 3** of the program.

A screenshot of a computer

Description automatically generated

**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.