2021

Morse Coder

GROUP PROJECT

TEAM 1

ADAM JOST | NEHA METLAPALLI

Morse Coder Group Project

Objective

The objective of the project was to build a "Morse Code tree" using character and Morse Code pairs stored in a data file, and then use the tree to encode and decode user messages.

Our Team

Our team consists of two Java developers:

- Adam Jost
- Neha Metlapalli

Team Member Contributions

Project report:

- Adam Jost
- Neha Metlapalli

UML Diagram:

- Adam Jost

Source Code (At Time of Report):

- *MorseCoderApp.java* was coded by Adam Jost
- *Layout.java* was coded by Adam Jost
- *MenuLayout.java* was coded by Adam Jost
- *EndecoderLayout.java* was coded by Adam Jost
- *ChatLayout.java* was coded by Adam Jost
- *ApplicationWindow.java* was coded by Adam Jost
- *Menu.java* was coded by Adam Jost
- *MorseEndecoder.java* was coded by Adam Jost
- *MorseChat.java* was coded by Adam Jost

- *MorseNode.java* was coded by Adam Jost
- *MorseCoder.java* was coded by Adam Jost
- *CustomButton.java* was coded by Neha Metlapalli
- *CustomIcon.java* was coded by Neha Metlapalli
- *MessageStrings.java* was coded by Adam Jost

Debugging:

- Neha Metlapalli
- Adam Jost

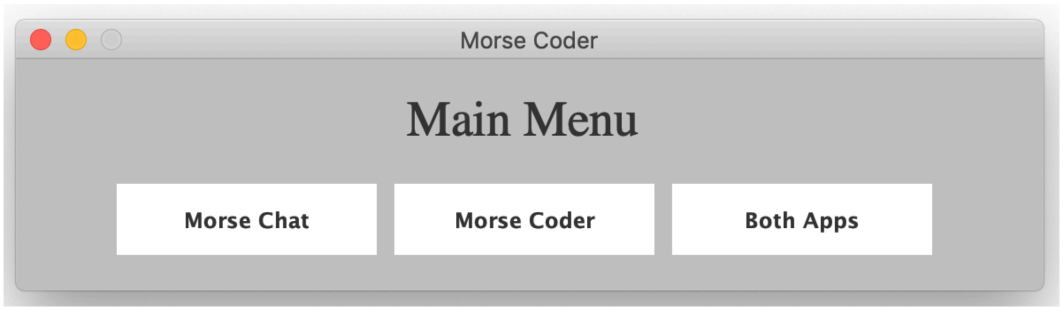
System Overview

A two-in-one, GUI based application consisting of a Morse Code chat application and a Morse Code encoding and decoding tool. Both items can be used independently or simultaneously. When used together a user can send and receive Morse Code messages in the Morse Code chat application and then use the decoding tool to decode and read the received messages.

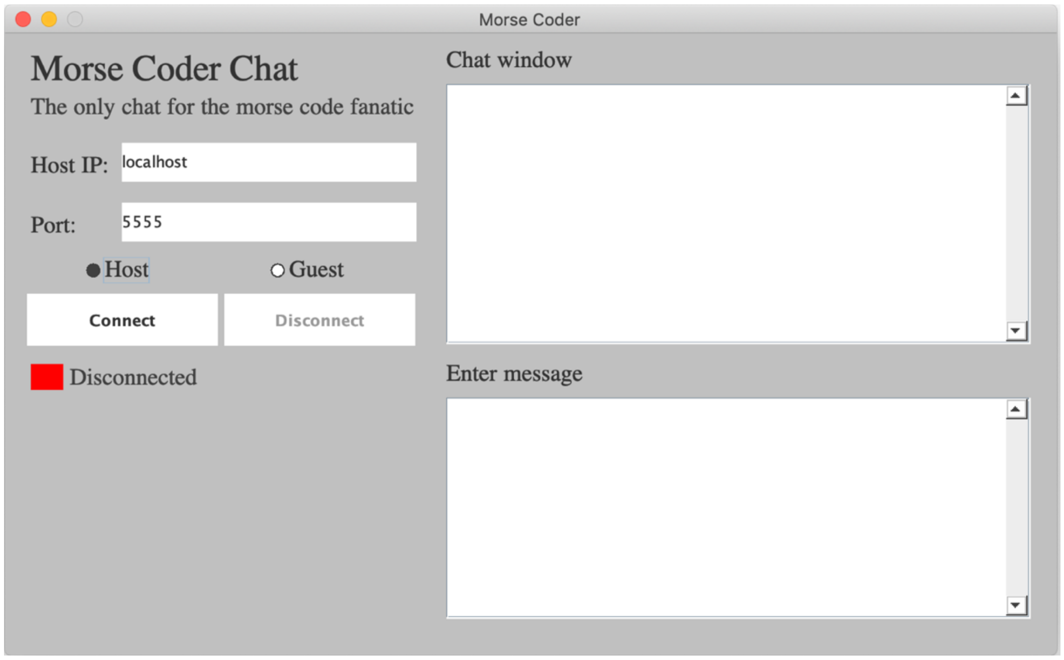
The Interface

The graphical user interface (GUI) for this application was built using *Swing* components. The *CrossPlatformLookAndFeel* class has been used to ensure that the GUI's components will look and behave the same across all platforms.

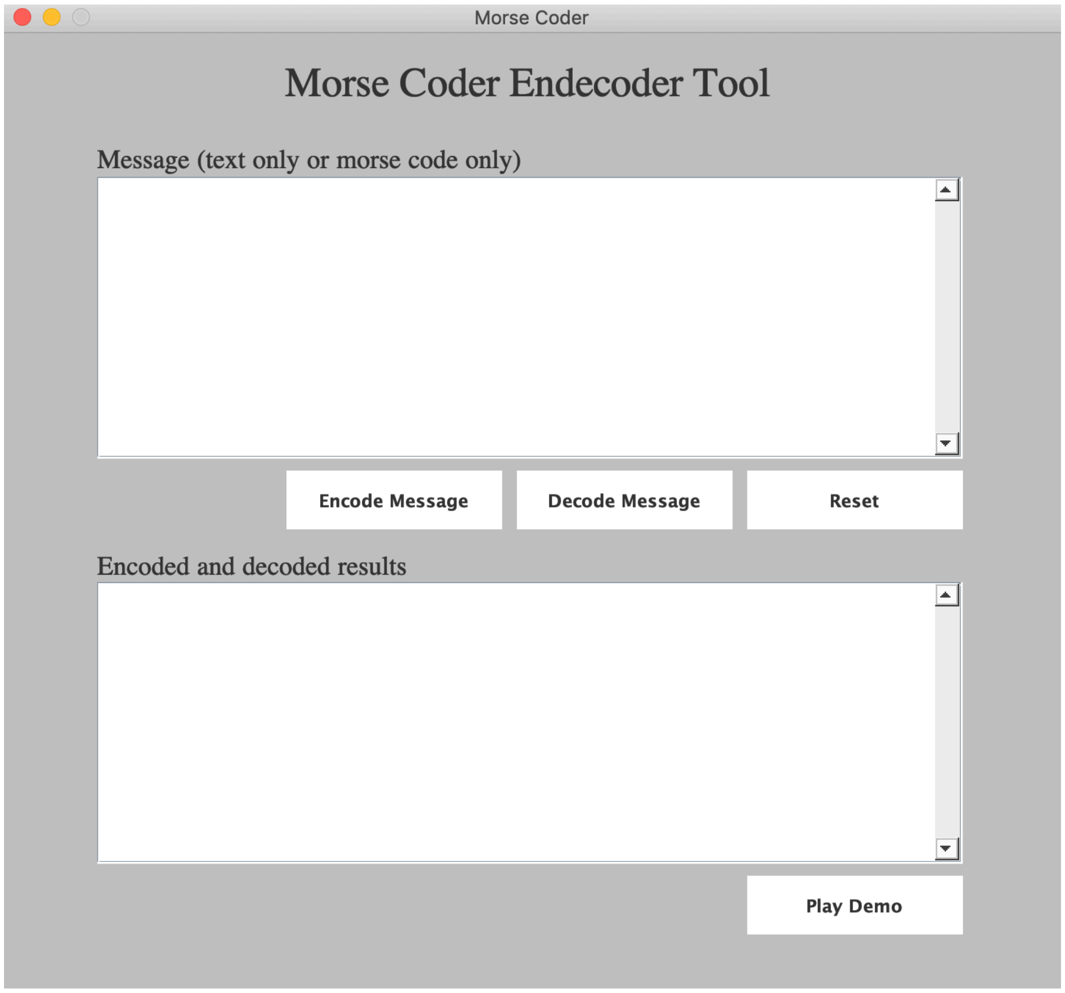
Menu



Morse Coder Chat



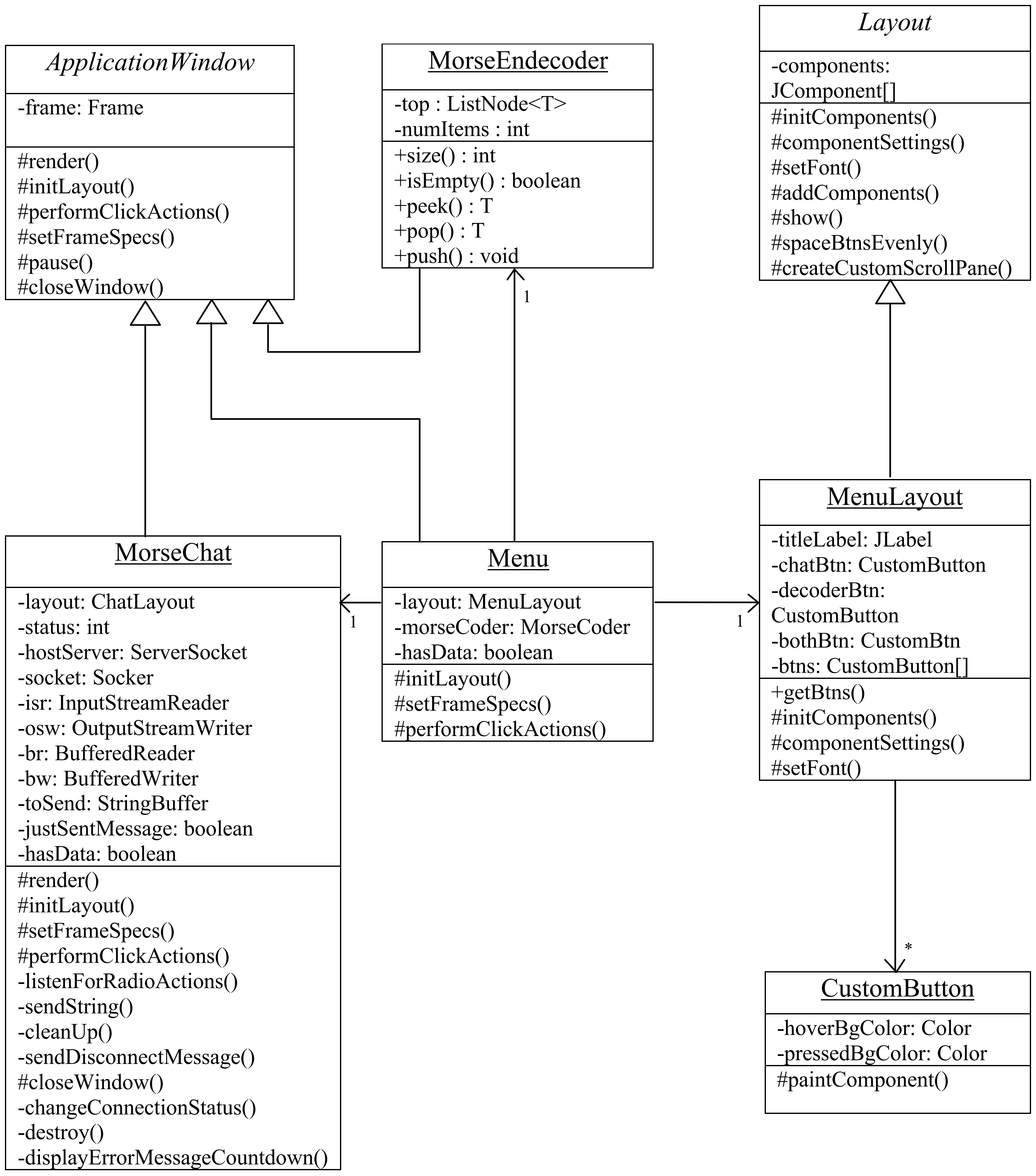
Morse Coder Endecoder


Morse Coder
Morse Coder Endecoder Tool
Message (text only or morse code only)
Encode Message
Decode Message
Reset
Encoded and decoded results
Play Demo

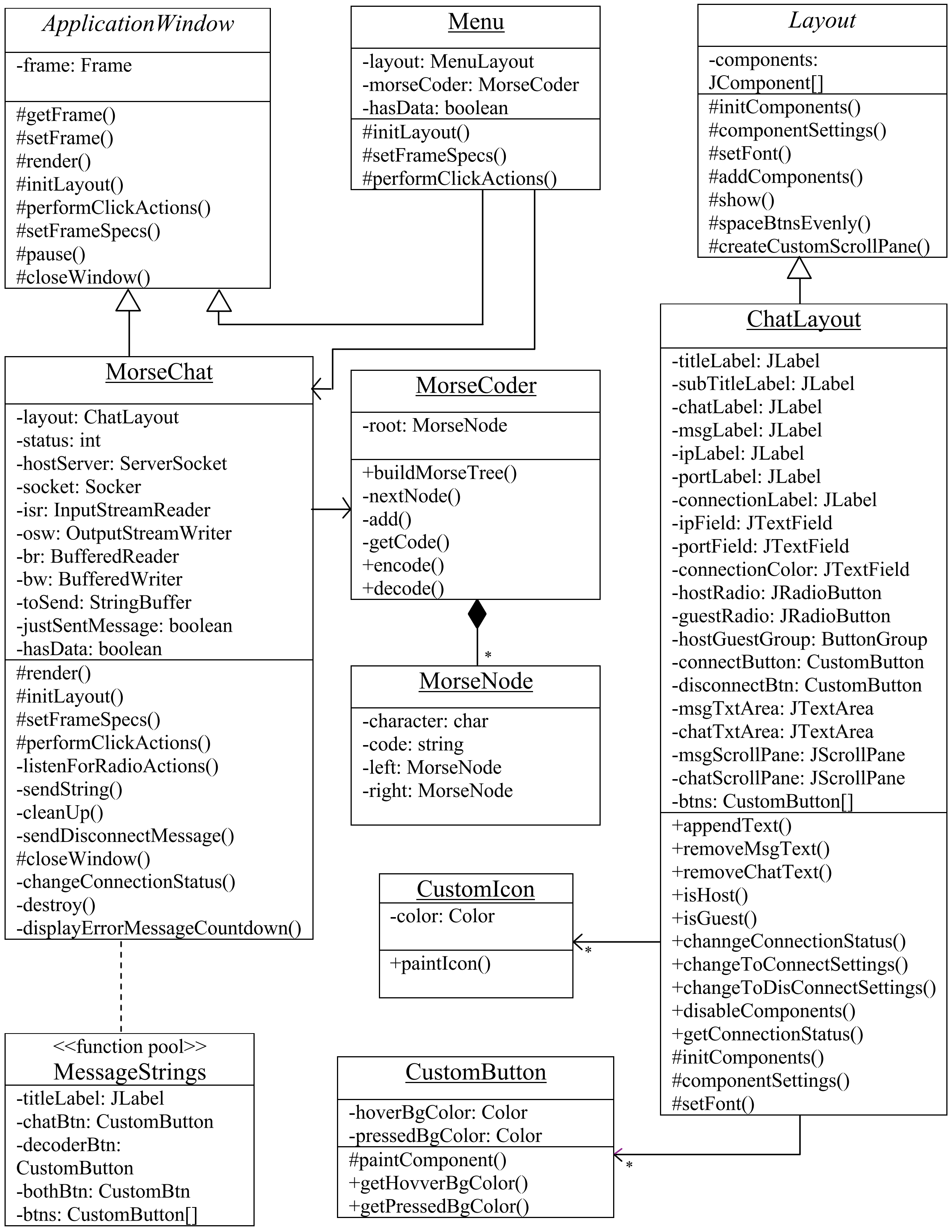
UML Diagrams

The diagram has been broken into sections due to size limitations.

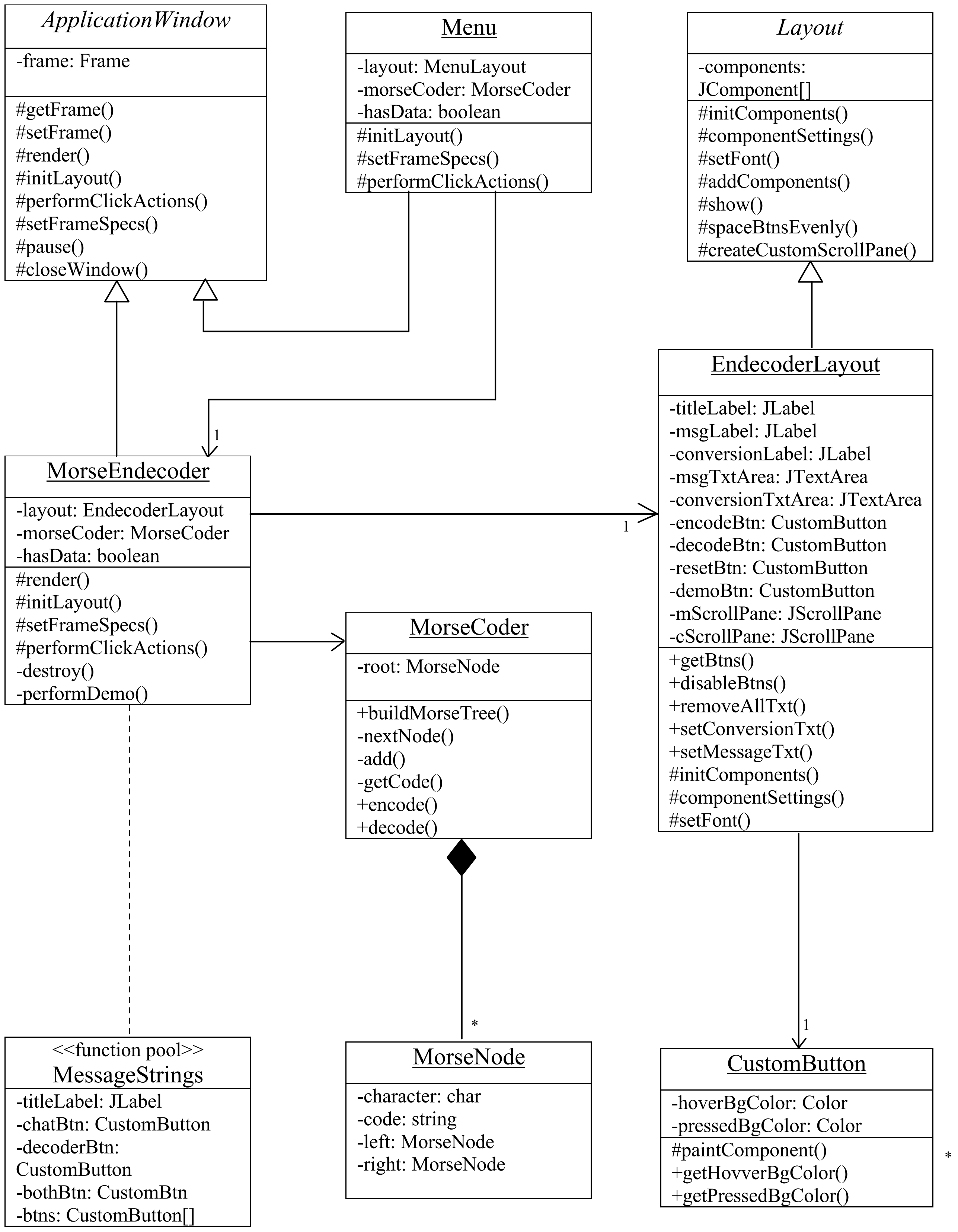
Menu Diagram



Morse Chat


ApplicationWindow
-frame: Frame
#getFrame()
#setFrame()
#render()
#initLayout()
#performClickActions()
#setFrameSpecs()
#pause()
#closeWindow()
Menu
-layout: MenuLayout
-morseCoder: MorseCoder
-hasData: boolean
#initLayout()
#setFrameSpecs()
#performClickActions()
Layout
-components: JComponent[]
#initComponents()
#componentSettings()
#setFont()
#addComponents()
#show()
#spaceBtnsEvenly()
#createCustomScrollPane()
MorseChat
-layout: ChatLayout
-status: int
-hostServer: ServerSocket
-socket: Socker
-isr: InputStreamReader
-osw: OutputStreamWriter
-br: BufferedReader
-bw: BufferedWriter
-toSend: StringBuffer
-justSentMessage: boolean
-hasData: boolean
#render()
#initLayout()
#setFrameSpecs()
#performClickActions()
-listenForRadioActions()
-sendString()
-cleanUp()
-sendDisconnectMessage()
#closeWindow()
-changeConnectionStatus()
-destroy()
-displayErrorMessageCountdown()
MorseCoder
-root: MorseNode
+buildMorseTree()
-nextNode()
-add()
-getCode()
+encode()
+decode()
*
MorseNode
-character: char
-code: string
-left: MorseNode
-right: MorseNode
ChatLayout
-titleLabel: JLabel
-subTitleLabel: JLabel
-chatLabel: JLabel
-msgLabel: JLabel
-ipLabel: JLabel
-portLabel: JLabel
-connectionLabel: JLabel
-ipField: JTextField
-portField: JTextField
-connectionColor: JTextField
-hostRadio: JRadioButton
-guestRadio: JRadioButton
-hostGuestGroup: ButtonGroup
-connectButton: CustomButton
-disconnectBtn: CustomButton
-msgTxtArea: JTextArea
-chatTxtArea: JTextArea
-msgScrollPane: JScrollPane
-chatScrollPane: JScrollPane
-btns: CustomButton[]
+appendText()
+removeMsgText()
+removeChatText()
+isHost()
+isGuest()
+channgeConnectionStatus()
+changeToConnectSettings()
+changeToDisConnectSettings()
+disableComponents()
+getConnectionStatus()
#initComponents()
#componentSettings()
#setFont()
CustomIcon
-color: Color
+paintIcon()
*
<<function pool>>
MessageStrings
-titleLabel: JLabel
-chatBtn: CustomButton
-decoderBtn: CustomButton
-bothBtn: CustomBtn
-btns: CustomButton[]
CustomButton
-hoverBgColor: Color
-pressedBgColor: Color
#paintComponent()
+getHovverBgColor()
+getPressedBgColor()
*

Morse Endecoder



System Design

Physical Design

Physical design relates to the actual input and output processes of the system such as how data is entered into a system, verified, processed, and displayed.

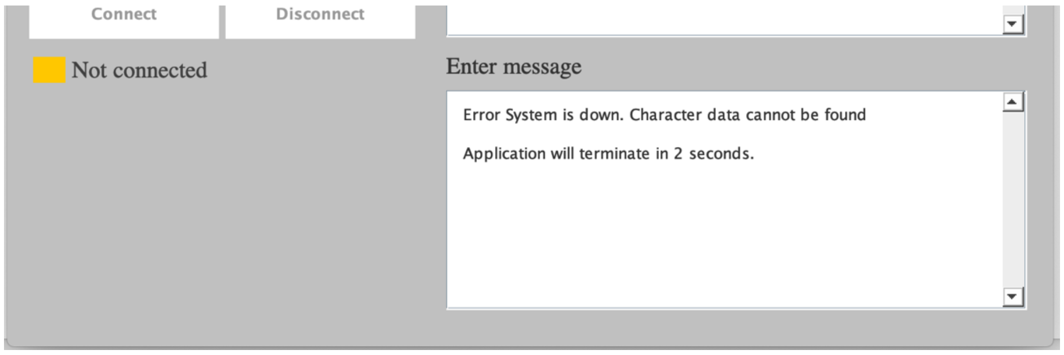
Input Design

File Input

At the start of the program, a one-time operation occurs where the system reads in a source document (.txt file) containing a list of all supported characters and their Morse Code representations. Each line in the input file must be either an empty line or a line consisting of a single character and its Morse Code representation.

Missing File Input

If in any event the input file is unable to be located, the system will disable all of the GUI's components and output an error message containing a ten second countdown informing the user that the system is down before closing the current application window.



Error System is down. Character data cannot be found

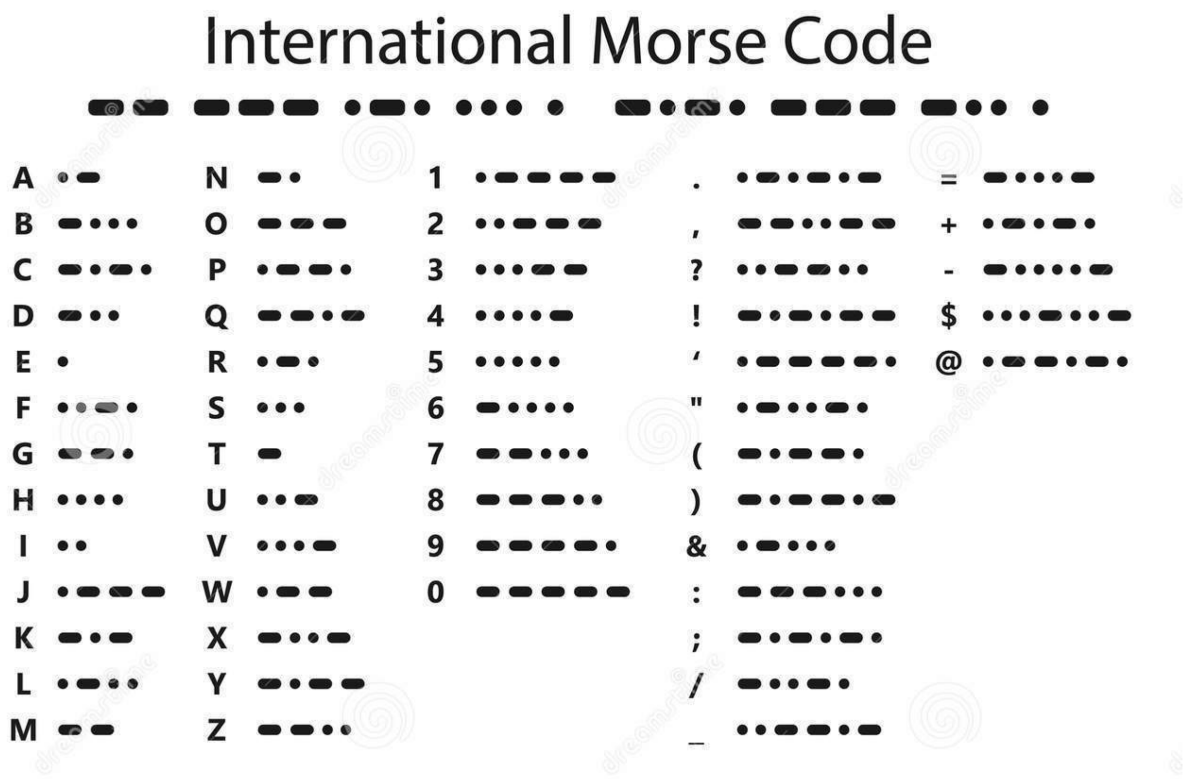
Application will terminate in 2 seconds.

User Input

In both application windows the user can enter a message into a JTextArea component in *string* format. Messages can consist of any combination of supported letters, digits, punctuation, and symbols.

Supported Characters and Morse Code

Image credit: Dreamstime.com



International Morse Code

−− −−− •−• ••• • −•−• −−− −•• •

A	•−	N	−•	1	•−−−−	.	•−•−•−	=	−•••−
B	−•••	O	−−−	2	••−−−	,	−−••−−	+	•−•−•
C	−•−•	P	•−−•	3	•••−−	?	••−−••	-	−••••−
D	−••	Q	−−•−	4	••••−	!	−•−•−−	$	•••−••−
E	•	R	•−•	5	•••••	'	•−−−−•	@	•−−•−•
F	••−•	S	•••	6	−••••	"	•−••−•		
G	−−•	T	−	7	−−•••	(	−•−−•		
H	••••	U	••−	8	−−−••	)	−•−−•−		
I	••	V	•••−	9	−−−−•	&	•−•••		
J	•−−−	W	•−−	0	−−−−−	:	−−−•••		
K	−•−	X	−••−			;	−•−•−•		
L	•−••	Y	−•−−			/	−••−•		
M	−−	Z	−−••			_	••−−•−		

Format of Morse Code Strings

The Morse code format is composed of four elements:

- Dot character
- Dash character
- Short gap (between letters) — one space long
- Medium gap (between words) — two spaces long

To illustrate, the phrase "MORSE CODE," written in Morse code format

-- --- .-. ... . -.-. --- -.. .

Valid Input

- **Valid Morse Code input**
    - Can only consist of '.' and '-' characters in supported Morse Code character combinations are considered valid.
        - .... . -.-- --..-- .-- .... .- - .----. ... ..- .--. ..--..
    - Can contain any amount of successive white space characters before, after, or within the Morse Code message. A single whitespace character signifies the end of a single character's code and two whitespace characters signify the end of a word.
        -   .-- .... .. - .   ... .--. .- -.-. .
    - Cannot contain letters, digits, punctuation or symbols. Messages containing these items will have these items removed during the encoding process in order to smartly convert them to valid input *strings*.
        -   1.) .—Even though this .... contains .. Morse Code – it is invalid because of . all of these letters, digits, and symbols. .- - ..- .--
    - Cannot contain unrecognized sequences of Morse Code characters.
        - ......-.....
        - ---------........-------
    - Cannot be an empty *string.*

Valid plain text input

- Can consist of any combination of supported letters, digits, punctuation, and symbols.

        - Hey, check this out! This is an example of valid input. Pretty cool isn't it?
    - Can have any amount of successive white space characters before, after, or within the plain text message.
        - Look   at   all these unnecessary   spaces   .
    - Cannot contain unsupported symbols (~`%^<>|*).
        - \#some ^symbols <are> |not| supported~\.
    - Cannot be an empty *string*.

Output

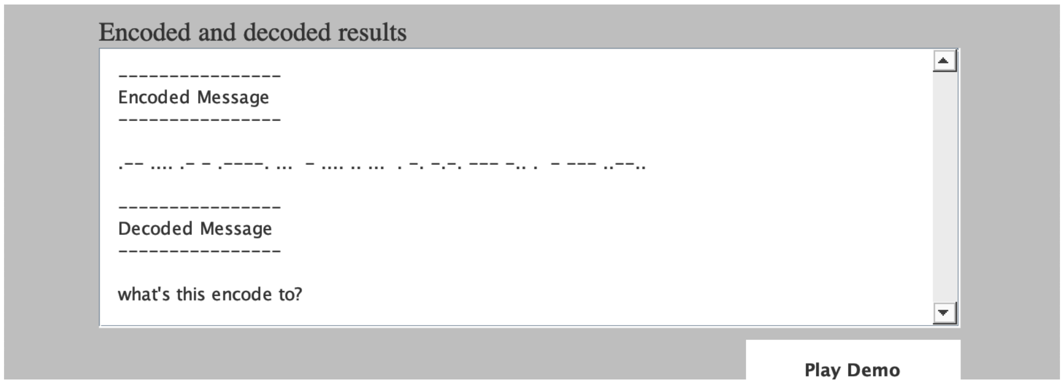
The output produced differs depending on the action being performed (e.g. encoding or decoding). The output also differs depending on which application window the input *string* is being entered into (i.e. Morse Chat or Morse Endecoder).

Morse Coder Endecoder - Encoding Output

When a user enters a valid, plain text *string* into the message *JTextArea* and then clicks the "Encode Message" button the system will encode the input message, and then decode the encoded message displaying both results to the user. In the event that multiple, successive whitespace characters are found during this process they will be reduced to a single space character (e.g. "   " will be reduced to ' '). If the input *string* is determined to be partially invalid the system will remove the unsupported characters from the message during the encoding process and will display a similar result as described above but with the addition of an error message and a list of all the invalid characters that were removed. If the input *string* was determined to consist of only unsupported characters the system will only output an error message stating unsupported characters exist in the input. Below are examples for each of the above identified situations.

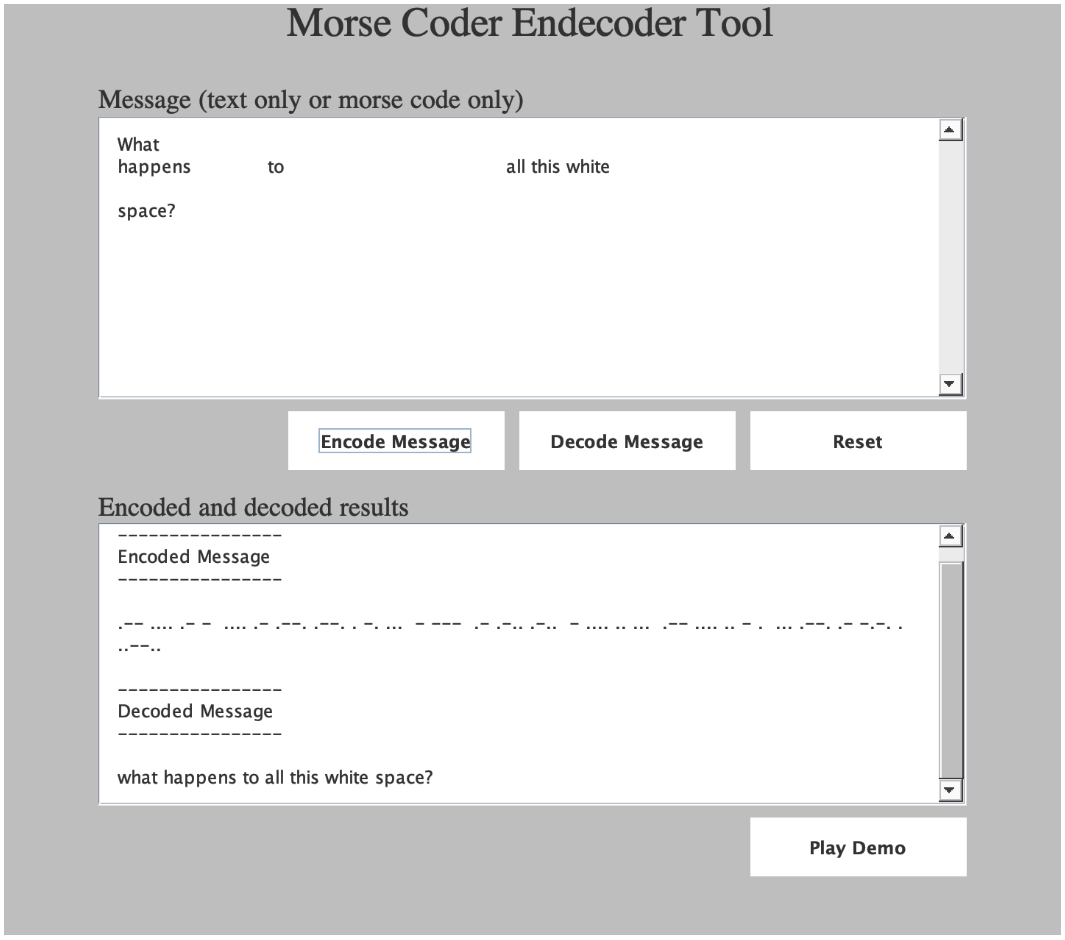
Output Resulting from Valid Input

Example One



```
----------------
Encoded Message
----------------

.-- .... .- - .----. ...  - .... .. ...  . -. -.-. --- -.. .  - --- ..--..

----------------
Decoded Message
----------------

what's this encode to?
```

Example Two

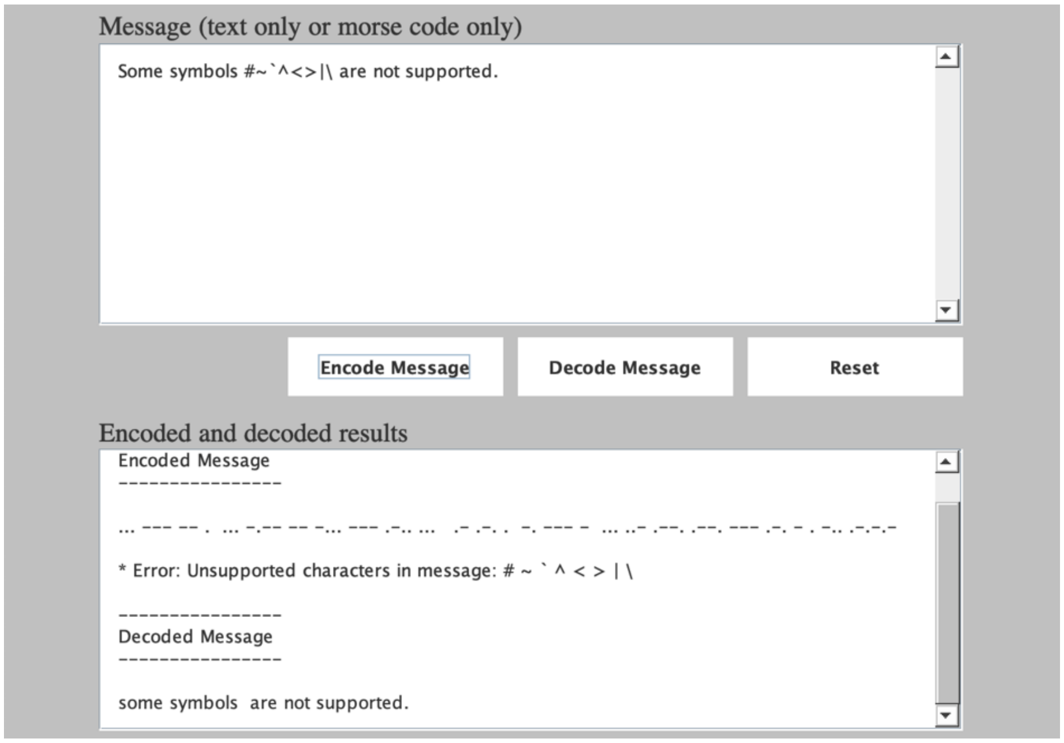

Morse Coder Endecoder Tool
Message (text only or morse code only)
Encode Message
Decode Message
Reset
Encoded and decoded results
Play Demo

```
What
happens          to                              all this white

space?
```

```
----------------
Encoded Message
----------------

.-- .... .- -  .... .- .--. .--. . -. ...  - ---  .- .-.. .-..  - .... .. ...  .-- .... .. - .  ... .--. .- -.-. .
..--..

----------------
Decoded Message
----------------

what happens to all this white space?
```

Output Resulting from Invalid Input

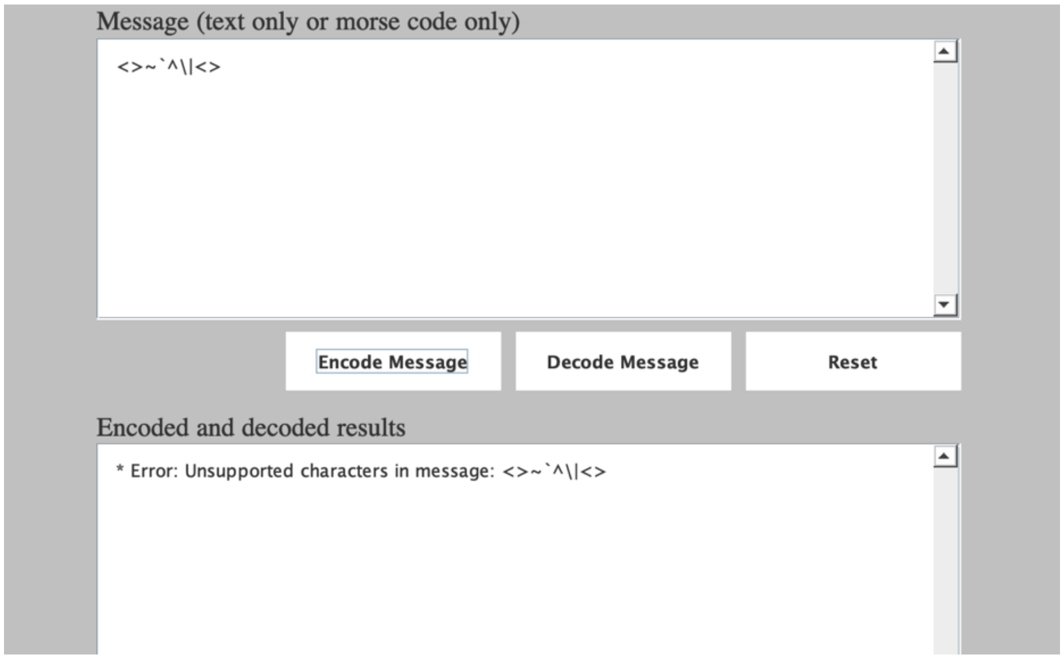
Example One


Message (text only or morse code only)
Encode Message
Decode Message
Reset
Encoded and decoded results

```
Some symbols #~`^<>|\ are not supported.
```

```
Encoded Message
----------------

... --- -- .  ... -.-- -- -... --- .-.. ...   .- .-. .  -. --- -  ... ..- .--. .--. --- .-. - . -.. .-.-.-

* Error: Unsupported characters in message: # ~ ` ^ < > | \

----------------
Decoded Message
----------------

some symbols  are not supported.
```

Example Two


Message (text only or morse code only)
Encode Message
Decode Message
Reset
Encoded and decoded results

```
<>~`^\|<>
```

```
* Error: Unsupported characters in message: <>~`^\|<>
```

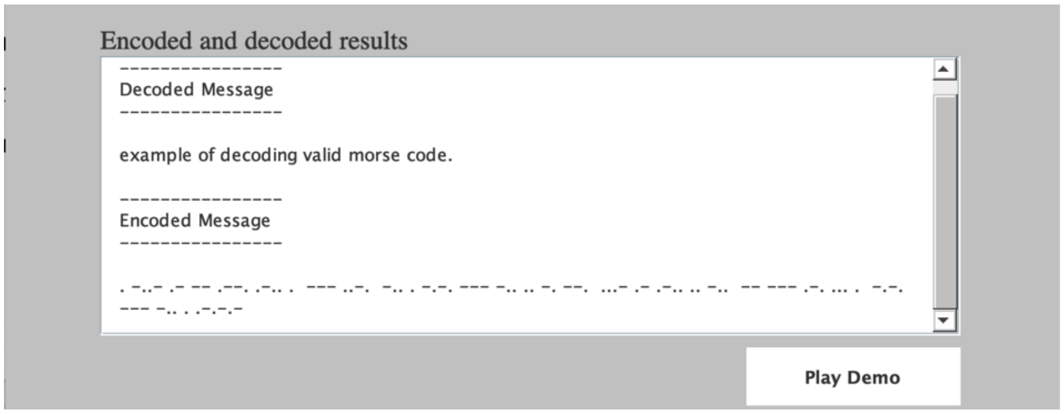
Morse Coder Endecoder – Decoding Output

When a user enters a valid, Morse Code *string* into the message *JTextArea* and then clicks the "Decode Message" button the system will decode the input message, and then encode the decoded message displaying both results to the user. In the event that more than two successive whitespace characters are found during this process they will be reduced to two space characters (e.g. "   " will be reduced to " "). This formatting process ensures proper spacing of words in the output *string*. If the input *string* is determined contain letters, digits, or symbols other than '.' or '-' the system will remove the invalid characters from the message during the decoding process and then display a similar result as described above but with the addition of an error message and a list of all the invalid characters that were removed. If the input *string* was determined to consist of only unsupported characters or partially the system will only output an error message stating that the system does not recognize the Morse Code. If the input *string* was determined to contain an invalid sequence of valid Morse Code characters that the system does not recognize the Morse Code. Below are examples for each of the above identified situations.

Output Resulting from Valid Input

Example One

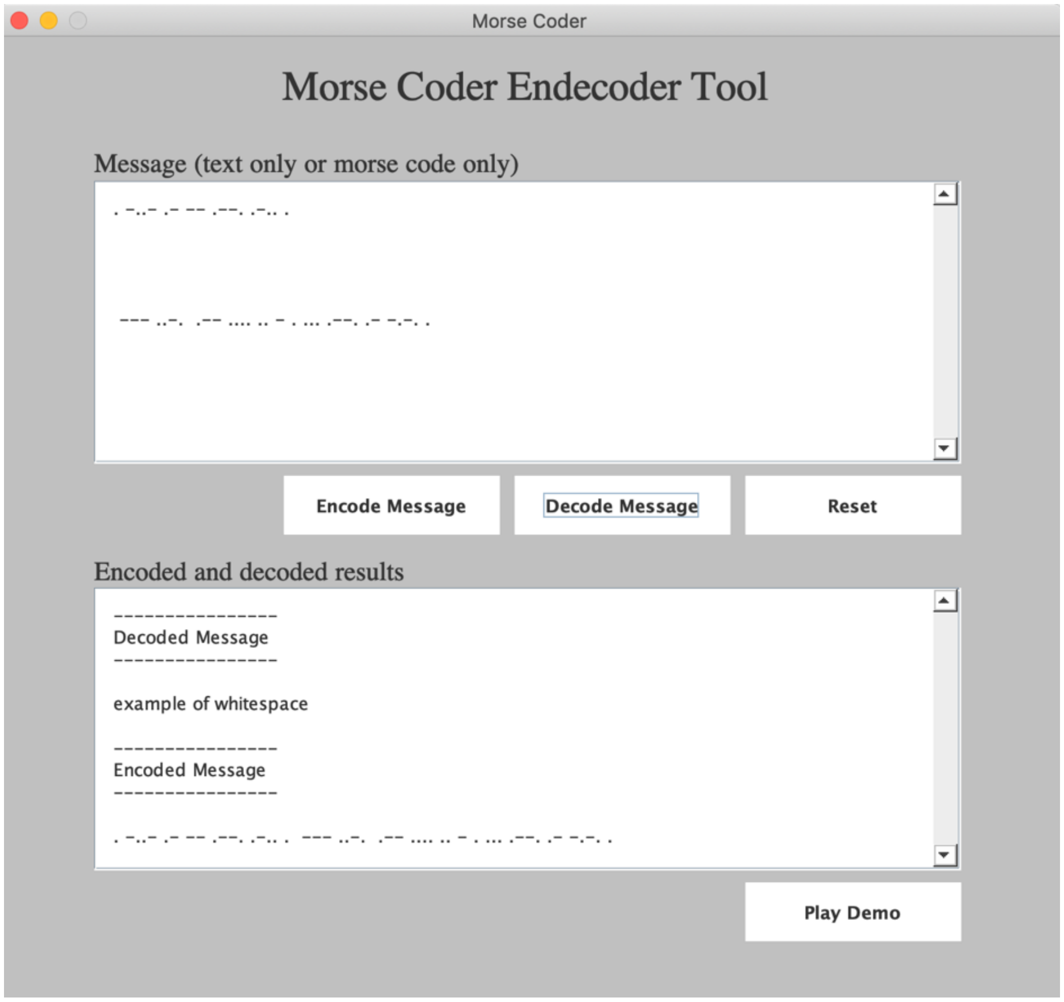
Valid Input



```
----------------
Decoded Message
----------------

example of decoding valid morse code.

----------------
Encoded Message
----------------

. -..- .- -- .--. .-.. .  --- ..-.  -.. . -.-. --- -.. .. -. --.  ...- .- .-.. .. -..  -- --- .-. ... .  -.-.
--- -.. . .-.-.-
```

Example Two

Valid Input – Removal of Extra Whitespace


Morse Coder
Morse Coder Endecoder Tool
Message (text only or morse code only)
Encode Message
Decode Message
Reset
Encoded and decoded results
Play Demo

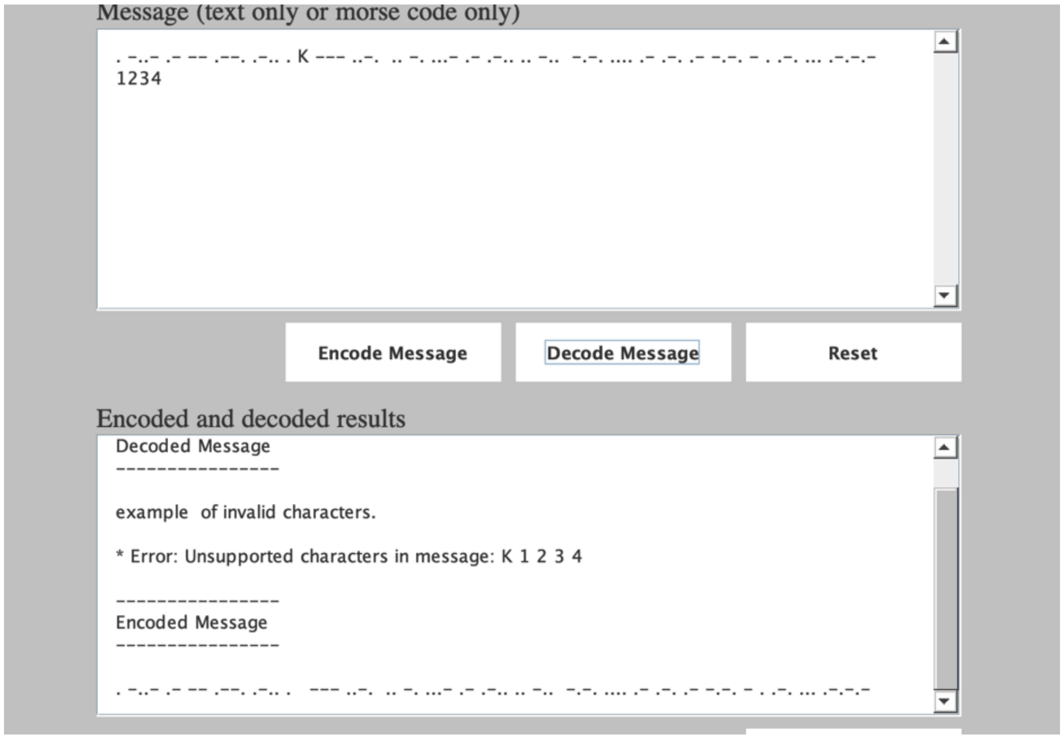
```
. -..- .- -- .--. .-.. .



 --- ..-.  .-- .... .. - . ... .--. .- -.-. .
```

```
----------------
Decoded Message
----------------

example of whitespace

----------------
Encoded Message
----------------

. -..- .- -- .--. .-.. .  --- ..-.  .-- .... .. - . ... .--. .- -.-. .
```

Output Resulting from Invalid Input

Example One

Input Containing Unsupported Characters

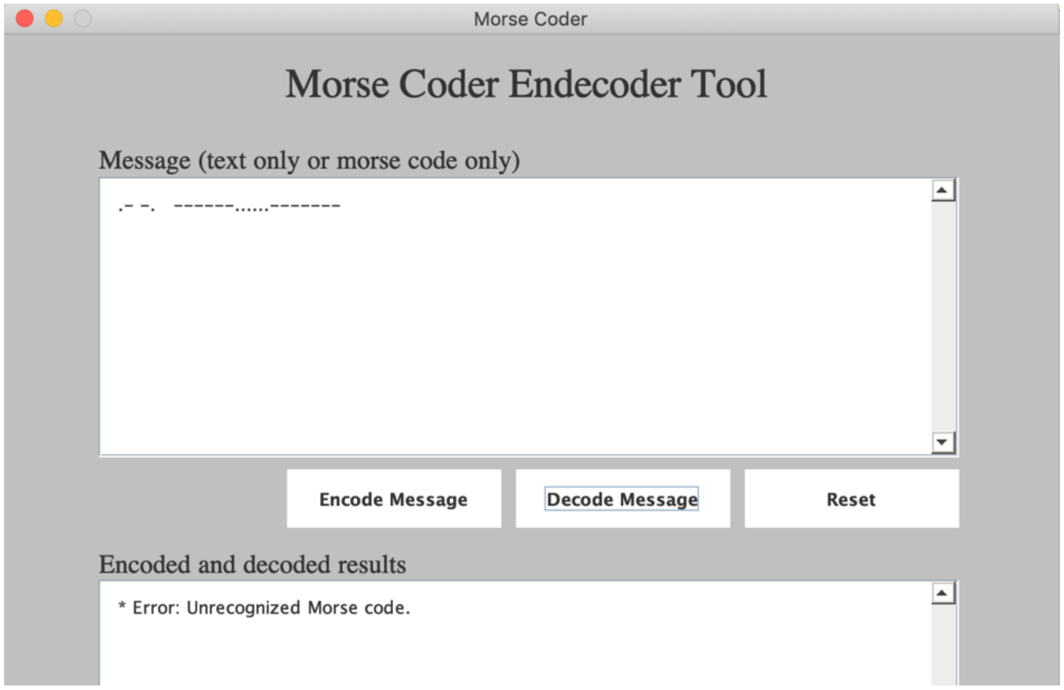

Message (text only or morse code only)
Encode Message
Decode Message
Reset
Encoded and decoded results

```
. -..- .- -- .--. .-.. . K --- ..-.  .. -. ...- .- .-.. .. -..  -.-. .... .- .-. .- -.-. - . .-. ... .-.-.-
1234
```

```
Decoded Message
----------------

example  of invalid characters.

* Error: Unsupported characters in message: K 1 2 3 4

----------------
Encoded Message
----------------

. -..- .- -- .--. .-.. .   --- ..-.  .. -. ...- .- .-.. .. -..  -.-. .... .- .-. .- -.-. - . .-. ... .-.-.-
```

Example Two

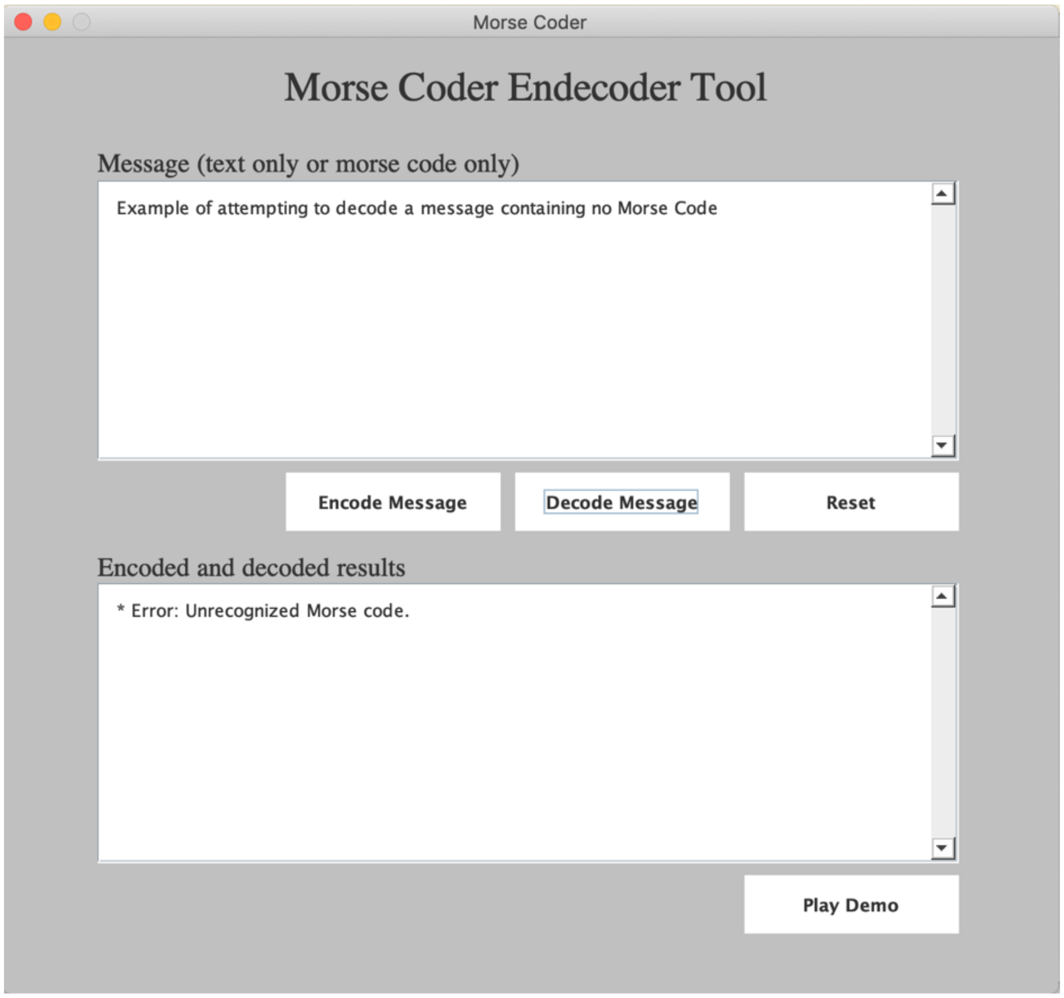
Invalid Morse Code character sequences


Morse Coder
Morse Coder Endecoder Tool
Message (text only or morse code only)
Encode Message
Decode Message
Reset
Encoded and decoded results

```
.- -.   ------......-------
```

```
* Error: Unrecognized Morse code.
```

Example Three


Morse Coder
Morse Coder Endecoder Tool
Message (text only or morse code only)
Encode Message
Decode Message
Reset
Encoded and decoded results
Play Demo

Example of attempting to decode a message containing no Morse Code

* Error: Unrecognized Morse code.

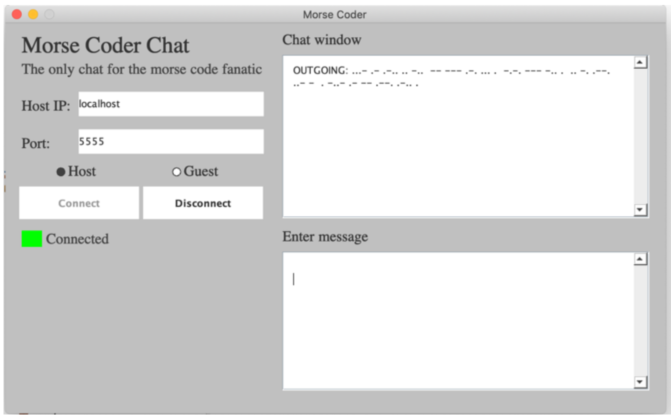
Morse Coder Chat - Output

The output for the Chat application will consist of incoming and outgoing messages written in Morse Code. Incoming messages will be prefixed by the *string* "Incoming: " and outgoing messages will be prefixed by the *string* "Outgoing: ". Input messages containing at least 80% Morse Code characters will simply be formatted to remove all extra whitespace , sent, and then outputted to both the sender and the receiver of the message. Input messages that consist of at least 80% Morse Code characters and also contain an invalid sequence of Morse Code characters or any non-Morse Code characters (e.g. letters, digits, punctuation, or symbols) will produce an error message that is only displayed to the user attempting to send the invalid message. Input messages consisting of 20% or less Morse Code characters will be determined to be plain text messages and will be formatted and encoded before being sent. Plain text messages containing any unsupported characters will produce an error message and will not be sent. Below are examples of each of the above-mentioned cases.

Output Resulting from Morse Code Input

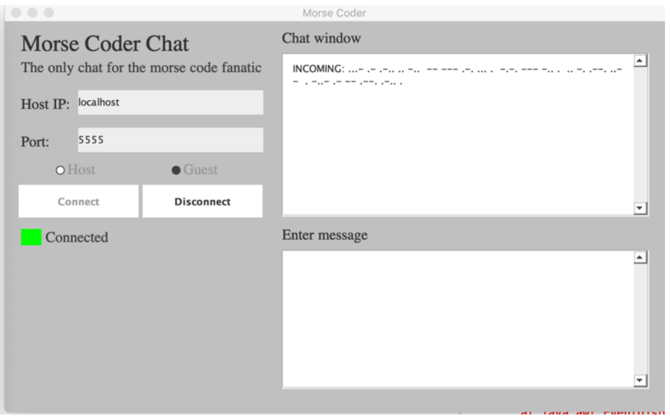
Example 1

Valid Input



Host IP: localhost
Port: 5555
● Host ○ Guest

```
OUTGOING: ...- .- .-.. .. -..  -- --- .-. ... .  -.-. --- -.. .  .. -. .--.
..- -  . -..- .- -- .--. .-.. .
```

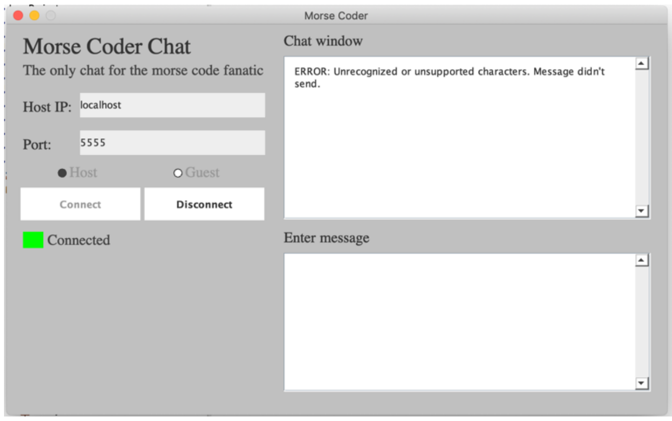


Host IP: localhost
Port: 5555
○ Host ● Guest

```
INCOMING: ...- .- .-.. .. -..  -- --- .-. ... .  -.-. --- -.. .  .. -. .--. ..-
-  . -..- .- -- .--. .-.. .
```

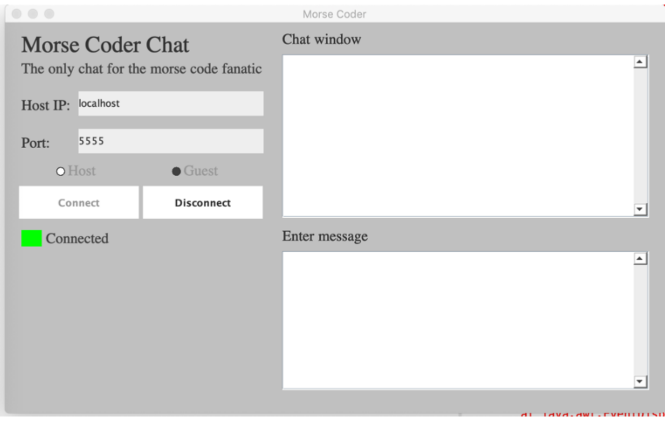
Example 2

Input Containing Invalid Characters



Host IP: localhost
Port: 5555
● Host ○ Guest

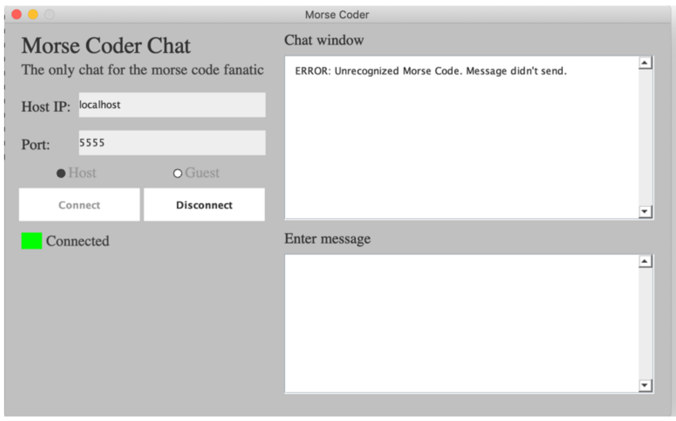
```
ERROR: Unrecognized or unsupported characters. Message didn't
send.
```



Host IP: localhost
Port: 5555
○ Host ● Guest

Example 3

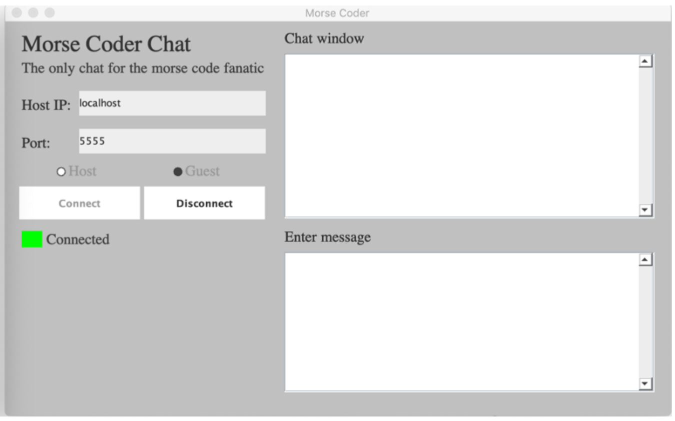
Invalid Morse Code Character Sequence


Morse Coder

Morse Coder Chat
The only chat for the morse code fanatic
Host IP: localhost
Port: 5555
● Host ○ Guest
Connect Disconnect
🟩 Connected

Chat window
ERROR: Unrecognized Morse Code. Message didn't send.

Enter message


Morse Coder

Morse Coder Chat
The only chat for the morse code fanatic
Host IP: localhost
Port: 5555
○ Host ● Guest
Connect Disconnect
🟩 Connected

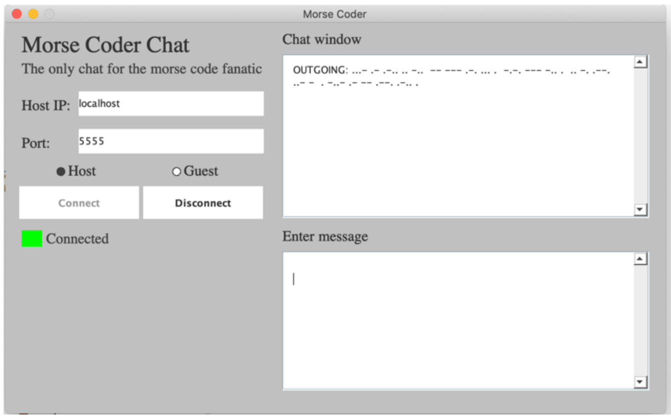
Chat window

Enter message

Output Resulting from Plain Text Input

Example 1

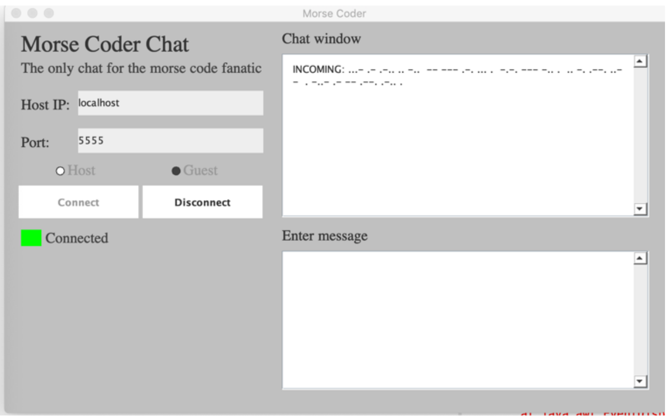
Valid Input


Morse Coder

Morse Coder Chat
The only chat for the morse code fanatic
Host IP: localhost
Port: 5555
● Host ○ Guest
Connect Disconnect
🟩 Connected

Chat window
```
OUTGOING: ...- .- .-.. .. -..  -- --- .-. ... .  -.-. --- -.. .  .. -. .--.
..- -  . -..- .- -- .--. .-.. .
```

Enter message


Morse Coder

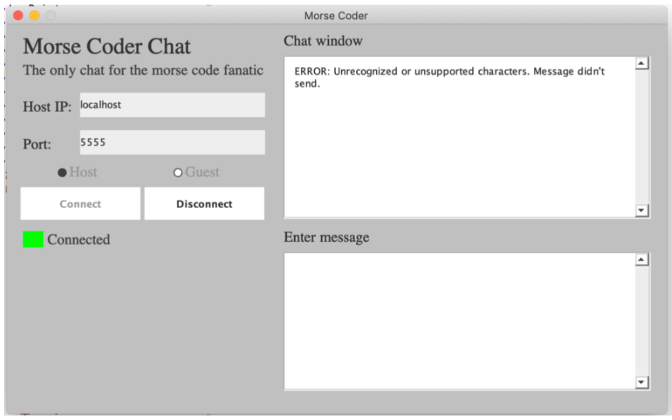
Morse Coder Chat
The only chat for the morse code fanatic
Host IP: localhost
Port: 5555
○ Host ● Guest
Connect Disconnect
🟩 Connected

Chat window
```
INCOMING: ...- .- .-.. .. -..  -- --- .-. ... .  -.-. --- -.. .  .. -. .--. ..-
-  . -..- .- -- .--. .-.. .
```

Enter message

Example 2

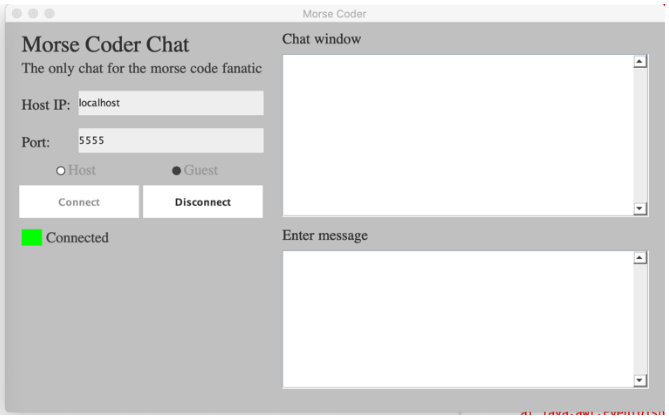
Input Containing Any Unsupported Characters


Morse Coder

Morse Coder Chat
The only chat for the morse code fanatic
Host IP: localhost
Port: 5555
● Host ○ Guest
Connect Disconnect
🟩 Connected

Chat window
ERROR: Unrecognized or unsupported characters. Message didn't send.

Enter message


Morse Coder

Morse Coder Chat
The only chat for the morse code fanatic
Host IP: localhost
Port: 5555
○ Host ● Guest
Connect Disconnect
🟩 Connected

Chat window

Enter message

Architectural Design

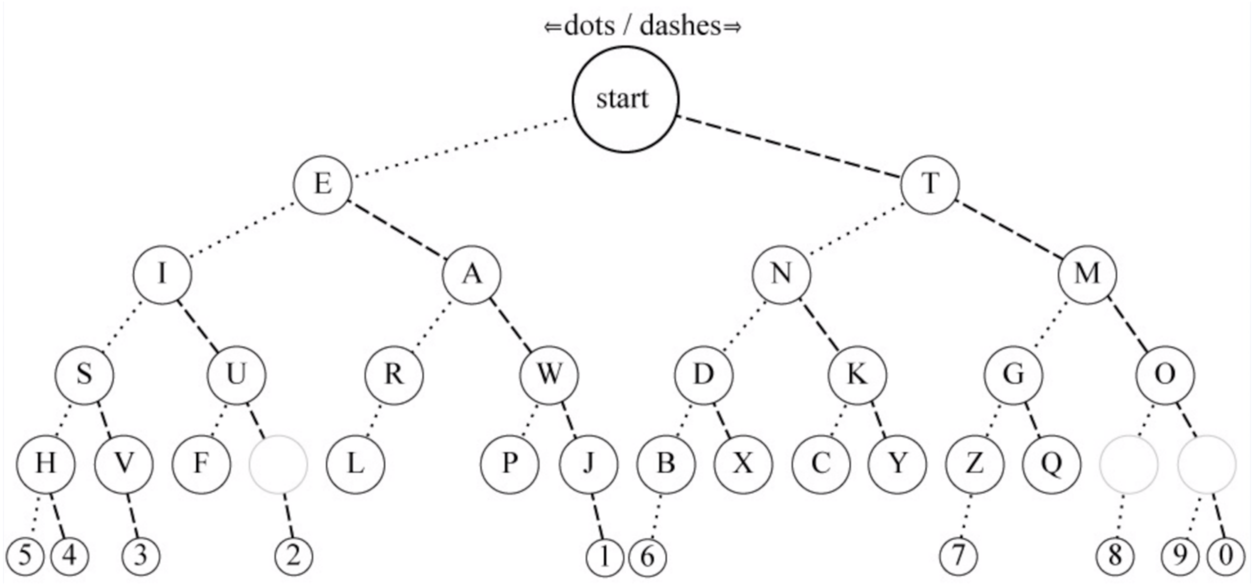
Architectural design focuses on the design of system architecture. It describes defines the structure and relationship between various parts of the systems.

Data Structures

- *MorseNode*
    - Used to store a single letter, digit, punctuation, or symbol character and its' Morse Code representation.
    - Used to implement the "*Morse Tree*".
- *Binary Tree ("Morse Tree")*
    - Implemented using **MorseNodes.*
    - Is a Java implementation of a *dichotomic search table allowing for easy and fast searching. For each encountered dot search the left subtree and for each dash search the right subtree.*

Morse Code Dichotomic Search Tree

Image credit: netninja.com



Major Classes

There is a total of 14 major classes:

- MorseCoderApp.java
- Layout.java
- MenuLayout.java
- EndecoderLayout.java
- ChatLayout.java
- ApplicationWindow.java
- Menu.java
- MorseEndecoder.java
- MorseChat.java
- MorseNode.java
- MorseCoder.java
- CustomButton.java
- CustomIcon.java
- MessageStrings.java

Relationship Between the Major Classes

- *MorseCoderApp* is the main program and creates a *Menu* and a *MorseCoder*.
- Menu extends *ApplicationWindow* and creates a *MenuLayout*.
- *MenuLayout* extends *Layout*.
- Menu creates several *CustomButtons*.
- When *CustomButtons* are clicked, Menu creates *MorseEndecoder* and/or *MorseChat*.
- *MorseChat* extends *ApplicationWindow* and creates a *ChatLayout*.
- *ChatLayout* extends *Layout*.
- *MorseEndecoder* extends *ApplicationWindow* and creates an *EndecoderLayout*.

- *MorseChat* and *MorseEndecoder* call functions from *MessageStrings* to format input *strings*.
- *MorseChat* and *MorseEndecoder* both use *MorseCoder* to encode and decode input *strings*.
- *MorseCoder* creates many *MorseNodes*.

Class Details

MorseCoderApp

- The “main program”.

MorseCoder

- Attributes:
    - *MorseNode root*: The root of the “Morse Tree”.
- Used to encode and decode input message *Strings*.
- Data Structures:
    - *MorseNode*: Used to implement a *Binary Tree*.
    - *Binary Tree (“Morse Tree”)*: Used for storing and searching for characters and their Morse Code representations when encoding and decoding input message *Strings*.

MorseNode

- Attributes:
    - *char character* – A letter, digit, punctuation or symbol character.
    - *String code* – The Morse Code representation of the character.
    - *MorseNode left* – The left child of the *MorseNode*.
    - *MorseNode right* – The right child of the *MorseNode*.
- Used to store a single character and Morse Code pair and to implement a *binary tree* data structure in *MorseCoder*.

Layout

- Abstract class.
- Extended by all layout classes.

MenuLayout

- Extends *Layout*.
- Creates many *Swing* components and adds them to a *Frame*.
- Methods manipulate the settings of the created components.

ChatLayout

- Extends *Layout*.
- Creates many *Swing* components and adds them to a *Frame*.
- Methods manipulate the settings of the created components.

EndecoderLayout

- Extends *Layout*.
- Creates many *Swing* components and adds them to a *Frame*.
- Methods manipulate the settings of the created components.

ApplicationWindow

- Abstract class.
- Extended by *Menu, MorseChat,* and *MorseEndecoder* classes.

Menu

- Extends *ApplicationWindow*.
- Creates and set specifications of a Frame.
- initializes a layout and adds it to the Frame.
- Adds *ActionListeners* to all of its layouts *CustomButton* components.

MorseChat

- Extends *ApplicationWindow*.
- Creates and set specifications of a Frame.
- initializes a layout and adds it to the Frame.
- Adds *ActionListeners* to all of its layouts *CustomButton and JRadio* components.
- Adds *KeyListeners* to the message *JTextArea*.
- Sends and receives input messages.

MorseEndecoder

- Extends *ApplicationWindow*.
- Creates and set specifications of a Frame.
- initializes a layout and adds it to the Frame.
- Adds *ActionListeners* to all of its layouts *CustomButton* components.
- Encodes and decodes input messages.

CustomButton

- Constructs a customized button with desired look.

CustomIcon

- Constructs a customized radio button with desired look.

MessageStrings

- *Function pool* consisting of several functions used to format user input *strings*.

Test Cases

Formatted headings will not be included in the expected result sections.

Test Case 1 : Morse Endecoder - Valid Plain Text Input String

This test was performed using the following input string
Morse Code

Expected Encoded Output

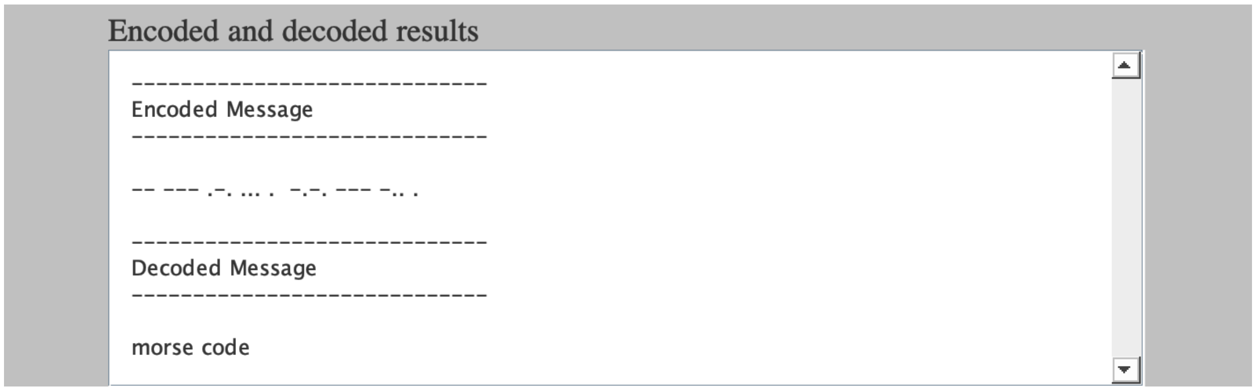
-- --- .-. ... . -.-. --- -.. .

Expected Decoded Output

morse code

Actual Results

The output results printed to the *JTextArea*


Encoded and decoded results

```
------------------------------
Encoded Message
------------------------------

-- --- .-. ... .  -.-. --- -.. .

------------------------------
Decoded Message
------------------------------

morse code
```

Differences: Expected vs. Actual Output

This test was a success. The correct output was printed as expected with no differences for both the encode and decode portions of the output.

Test Case 2 : Morse Endecoder – Invalid Morse Code Sequence

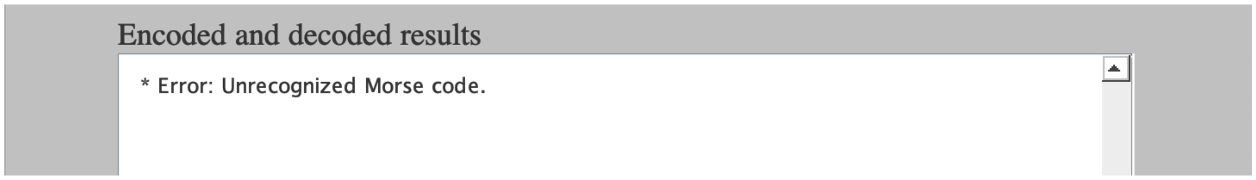
This test was performed using the following input string

------.......-------

Expected Output

* Error: Unrecognized Morse code.

Actual Results
The output results printed to the *JTextArea*



```
* Error: Unrecognized Morse code.
```

Differences: Expected vs. Actual Output

This test was a success. The system outputted the correct error message to the *JTextArea* as expected.

Future Improvements

This application can be improved in many ways. First improvement would be to make the layouts and their components responsive to allow window resizing. Next, would be to include the support of Morse code prosigns, abbreviations, and accent letters. A final way to improve this application would be to allow users to connect the Morse Chat to other users outside of their *Local Area Network (LAN)*.