

RMIT University

School of Engineering

EEET2248 – Electrical Engineering Analysis

Lab Experiment #2

Resistor Colour Decoder

Lecturer: Dr. Katrina Neville

Student Name: Nathan Paul Williams

Student Number: s3707244

Group: Tuesday 10:30-12:30

Submission Due Date: 27th April 2018

**Problem Statement**

Create a program (featuring a user-defined function and a GUI) that can decode the colour bands on a 4 or 5 band resistor and output data (resistance and tolerance) in a user-friendly format

**Input Data**

* Colours of each band
* Number of bands (4 or 5)

**Output Data**

* Resistance
* Tolerance
* Resistance Range
* Illustration of Resistor

**Design**

Figure : Resistor Colour Decoder Flowchart

Drop Down Menu Input

Nargin

Calculate Button

Resistor Decoder Function

(4 inputs)

Resistor Decoder Function

(5 inputs)

Print Result String with Output Variables

Call Painter Function to decode input to colours

Print Illustration

4 Inputs

5 Inputs

Output

Output

For my project design I decided to implement a Graphical User Interface (GUI) tod make the program much more user friendly. My GUI would feature 5 drop down menus with colour options representing the colours on the resistor the user was attempting to decode. If it was only a 4 band resistor the user would be asked to leave the 3rd drop down on the option ‘empty’ as this band represents the 3rd digit of the resistance in the case of a 5 band resistor, hence not relevant. Once the correct drop-down options were selected, pressing a button would trigger operations based on the input data. This included calling a user defined function to output resistance, tolerance, range and a character that correctly represented the magnitude of the resistance (e.g. ‘K’ representing 1000x the number to make it more user-friendly to interpret). Once the function returned its output, a simple string would be printed in a textbox using ‘sprintf’ that printed out the results featuring the relevant variables (e.g. ‘This resistor has a resistance of x variables ….’).

Once this is printed another user-defined function is called that translates the drop down menu inputs to colours (in the form of a 1x3 matrix representing RGB). With the output from this function an illustration is “drawn” to represent the input resistor bands which the user can compare to the physical resistor they are holding to avoid input errors. A “Help” button is also implemented that when pressed will open up a new app/window detailing further instructions/explanation of how to read the resistor and input the data. The user can also ask for more info which will open a webpage on the topic.

This design is centred around being user-friendly by avoiding possible errors, providing assistance and making the results as easily interpretable as possible. The flow chart in figure 1 illustrates the basic operations of the program from start to end.

A screenshot of a cell phone

Description generated with high confidence**Output**

Figure : GUI Example Output

A screenshot of a cell phone

Description generated with very high confidence

Figure : Help Window

**Testing**

To test the user-defined function another script was written that imported an excel spreadsheet containing random values and the expected output resistance and tolerance if they were to represent the values of an input resistor. The script then used a while loop to input the random values into the ResistorFunction and compare the output to the expected values from the spreadsheet. The script is as follows:

clc;

clear;

num = xlsread("ResistorFunctionTestSheet.xlsx");

k = 1;

correct = 0;

total = 0;

while(num(k,1) ~= 888)

if (num(k,3) == 888)

[r,t,rmin,rmax,rstring] = ResistorFunction(num(k,1),num(k,2),num(k,4),num(k,5));

if (rstring == 'K')

r = r\*1000;

elseif (rstring == 'M')

r = r\*1000000;

end

if ((r == num(k,6)) && (t == num(k,7)))

correct = correct + 1;

end

end

if (num(k,3) ~= 888)

[r,t,rmin,rmax,rstring] = ResistorFunction(num(k,1),num(k,2),num(k,3),num(k,4),num(k,5));

if (rstring == 'K')

r = r\*1000;

elseif (rstring == 'M')

r = r\*1000000;

end

if ((r == num(k,6)) && (t == num(k,7)))

correct = correct + 1;

end

end

total = total + 1;

k = k + 1;

end

fprintf('Resistor function correctly tested %g out of %g inputs\n',correct,total);

When run this script gives output *“Resistor function correctly tested 60 out of 60 inputs”* which shows that all of the tests come out correct providing evidence that our ResistorFunction is working correctly.

Through running the program the painter function appears to be working correctly, printing all 12 programmed colours. This appears to work correctly for all bands (1-5). The help button also correctly opens a new app/window.

Three tests were then ran with random input to test the GUI function:

1. Resistor: Red, Blue, Orange, Green, Gold

Hand working: Resistance = 263 \* 100000 = 26.3M, Tolerance = 5%, Range = (26.3M +/- 1.315M)

A screenshot of a cell phone

Description generated with high confidence

Figure : Test1 Output

1. Resistor: White, Grey, Yellow, Silver

Hand working: Resistance = 098 \* 10000 = 980K, Tolerance = 10%, Range = 980K +/- 98K

A screenshot of a cell phone

Description generated with very high confidence

Figure : Test2 Output

1. Resistor: Black, Black, Black, Brown (GUI default state)

Hand working: Resistance = 000\*1 = 0, Tolerance = 1%, Range = 0 +/- 0

A screenshot of a cell phone

Description generated with very high confidence

Figure : Test3 Output (GUI default)

The three tests ran showed that the GUI output agreed with the hand working exactly. The hand working was based on the chart provided [1]. This is evidence that our program is accurate and outputting the data as intended.

**Discussion**

A GUI was implemented using app designer as this simplified the input process for the user and avoided a lot of errors possible with a text-based input system. With the GUI it allowed me to easily assign values to the colours selected in the drop-down menus. This allowed me to simplify my algorithms substantially. The user-defined function could then be passed simple integers which it then performs some arithmetic on to output the required variables. The GUI then uses these variables to display the desired output.

My program allows for 4 or 5 band input by instructing the user to leave the 3rd drop down as empty in the case of a 4 band resistor. Alternative designs were considered but to have the correct drop down menus for a 4 band it would require an extra button and drop down forcing the user to select if the resistor was a 4 or 5 band. I decided that this extra process of selecting and pushing the button caused more work than what I deemed worth the extra “user-friendliness” if applied.

**Extra Features:**

* **Results Window:** The results window uses ‘sprintf()’ to print a string in the results text box that neatly shows the resistance, tolerance and range. This was applied to make the results as easy as possible to interpret for the user. It also outputs the resistance values in the ‘eng’ form using a character ‘K’ or ‘M’ (e.g. 198000 Ohms is printed as 198K Ω).
* **Illustrator:** The illustrator will colour in 5 boxes according to what has been input in the drop-down menus. This illustration is supposed to represent the input resistor so the user can visually verify that they have input their resistor correctly.
* **Help Button:** The help function when called willed open a new app which features a text box and 2 buttons as shown in figure 3. This text box contains some further instructions/information regarding use of the program. The “I understand” button will simply close the app and return to the original GUI. The “More Info” button will open a webpage [2] provides further information on the resistor colour code.

**Conclusion**

A solution has been implemented that allows the user to decode the properties of a resistor by inputting the colours of the bands displayed. The solution uses a GUI and other extra features to ensure the input and output process is user friendly, minimises errors and easy to interpret. The program makes use of a user defined function that can interpret the both 4 and 5 band resistors. In conclusion, the program meets the demands of the problem statement and provides a suitable solution however; could require further testing to ensure maximum accuracy and minimal possible errors.

**Note: To run app please run file “ResistorColourDecoderApp.mlapp” with all files in the current directory**

**References**

[1] Physics and Radio Electronics. (2013, 24/04/2018). Resistor Color Code. Available: <http://www.physics-and-radio-electronics.com/electronic-devices-and-circuits/passive-components/resistors/resistorcolorcode.html>

[2] Stack Exchange. (2010, 24/04/2018). Colour Coding. Available: <https://electronics.stackexchange.com/tags/colour-coding/info>

**Appendix (Matlab Script)**

**Resistor Function**

function [r,t,rmin,rmax,rstring] = ResistorFunction(a,b,c,d,e)

if nargin == 4

r = ((a\*10)+b) \* c;

t = d;

end

if nargin == 5

r = ((a\*100)+(b\*10)+c)\*d;

t = e;

end

rstring = '';

if (r >= 1000)

r = r/1000;

rstring = 'K';

end

if (r >= 1000)

r = r/1000;

rstring = 'M';

end

range = (r/100) \* t;

rmin = r - range;

rmax = r + range;

%rmin = round(rmin);

%rmax = round(rmax);

End

**Resistor Colour Decoder App**

classdef ResistorColourDecoderApp < matlab.apps.AppBase

% Properties that correspond to app components

properties (Access = public)

UIFigure matlab.ui.Figure

Band1DropDownLabel matlab.ui.control.Label

Band1DropDown matlab.ui.control.DropDown

Band2DropDownLabel matlab.ui.control.Label

Band2DropDown matlab.ui.control.DropDown

Band3DropDownLabel matlab.ui.control.Label

Band3DropDown matlab.ui.control.DropDown

Band4DropDownLabel matlab.ui.control.Label

Band4DropDown matlab.ui.control.DropDown

Band5DropDownLabel matlab.ui.control.Label

Band5DropDown matlab.ui.control.DropDown

ResistorColourCodeDecoderLabel matlab.ui.control.Label

CalculateButton matlab.ui.control.Button

NOTEIfresistoronlyhas4bandsleaveBand3asEmptyLabel matlab.ui.control.Label

EditField matlab.ui.control.EditField

EditField\_2 matlab.ui.control.EditField

EditField\_3 matlab.ui.control.EditField

EditField\_4 matlab.ui.control.EditField

EditField\_5 matlab.ui.control.EditField

HelpButton matlab.ui.control.Button

ResultsLabel matlab.ui.control.Label

InputLabel matlab.ui.control.Label

ResistorIllustrationLabel matlab.ui.control.Label

ForhelppleaseclickbelowLabel matlab.ui.control.Label

ResultText matlab.ui.control.TextArea

end

methods (Access = private)

% Button pushed function: CalculateButton

function CalculateButtonPushed(app, event)

%convert drop down values to doubles

band1 = str2double(app.Band1DropDown.Value);

band2 = str2double(app.Band2DropDown.Value);

if(app.Band3DropDown.Value ~= 'empty')

band3 = str2double(app.Band3DropDown.Value);

else

band3 = app.Band3DropDown.Value;

end

band4 = str2double(app.Band4DropDown.Value);

band5 = str2double(app.Band5DropDown.Value);

%input values into function

if (band3 == 'empty')

[R,T,Tmin,Tmax,RStr] = ResistorFunction(band1,band2,band4,band5);

else

[R,T,Tmin,Tmax,RStr] = ResistorFunction(band1,band2,band3,band4,band5);

end

TextOutput = sprintf('This resistor has a resistance of %g%s Ω with a tolerance of %g %%.\nThis gives it a range of %g%s Ω to %g%s Ω.',R,RStr,T,Tmin,RStr,Tmax,RStr);

%'print' final value

%app.ResistanceEditField.Value = R;

%app.ToleranceEditField.Value = T;

%app.minRangeEditField.Value = Tmin;

%app.maxRangeEditField.Value = Tmax;

app.ResultText.Value = TextOutput

%draw diagram

app.EditField.BackgroundColor = Painter(app.Band1DropDown.Value, 'digits');

app.EditField\_2.BackgroundColor = Painter(app.Band2DropDown.Value, 'digits');

if (app.Band3DropDown.Value ~= 'empty')

app.EditField\_3.BackgroundColor = Painter(app.Band3DropDown.Value, 'digits');

else

app.EditField\_3.BackgroundColor = [0.94 0.94 0.94]

end

app.EditField\_4.BackgroundColor = Painter(app.Band4DropDown.Value, 'multiplier');

app.EditField\_5.BackgroundColor = Painter(app.Band5DropDown.Value, 'tolerance');

end

% Button pushed function: HelpButton

function HelpButtonPushed(app, event)

Help;

end

end

% App initialization and construction

methods (Access = private)

% Create UIFigure and components

function createComponents(app)

% Create UIFigure

app.UIFigure = uifigure;

app.UIFigure.Position = [100 100 640 480];

app.UIFigure.Name = 'UI Figure';

% Create Band1DropDownLabel

app.Band1DropDownLabel = uilabel(app.UIFigure);

app.Band1DropDownLabel.BackgroundColor = [0.9412 0.9412 0.9412];

app.Band1DropDownLabel.HorizontalAlignment = 'right';

app.Band1DropDownLabel.Position = [89 370 44 15];

app.Band1DropDownLabel.Text = 'Band 1';

% Create Band1DropDown

app.Band1DropDown = uidropdown(app.UIFigure);

app.Band1DropDown.Items = {'Black', 'Brown', 'Red', 'Orange', 'Yellow', 'Green', 'Blue', 'Violet', 'Grey', 'White'};

app.Band1DropDown.ItemsData = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'};

app.Band1DropDown.BackgroundColor = [0.9412 0.9412 0.9412];

app.Band1DropDown.Position = [148 366 100 22];

app.Band1DropDown.Value = '0';

% Create Band2DropDownLabel

app.Band2DropDownLabel = uilabel(app.UIFigure);

app.Band2DropDownLabel.HorizontalAlignment = 'right';

app.Band2DropDownLabel.Position = [89 330 44 15];

app.Band2DropDownLabel.Text = 'Band 2';

% Create Band2DropDown

app.Band2DropDown = uidropdown(app.UIFigure);

app.Band2DropDown.Items = {'Black', 'Brown', 'Red', 'Orange', 'Yellow', 'Green', 'Blue', 'Violet', 'Grey', 'White'};

app.Band2DropDown.ItemsData = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'};

app.Band2DropDown.Position = [148 326 100 22];

app.Band2DropDown.Value = '0';

% Create Band3DropDownLabel

app.Band3DropDownLabel = uilabel(app.UIFigure);

app.Band3DropDownLabel.HorizontalAlignment = 'right';

app.Band3DropDownLabel.Position = [89 289 44 15];

app.Band3DropDownLabel.Text = 'Band 3';

% Create Band3DropDown

app.Band3DropDown = uidropdown(app.UIFigure);

app.Band3DropDown.Items = {'Empty', 'Black', 'Brown', 'Red', 'Orange', 'Yellow', 'Green', 'Blue', 'Violet', 'Grey', 'White'};

app.Band3DropDown.ItemsData = {'empty', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'};

app.Band3DropDown.Position = [148 285 100 22];

app.Band3DropDown.Value = 'empty';

% Create Band4DropDownLabel

app.Band4DropDownLabel = uilabel(app.UIFigure);

app.Band4DropDownLabel.HorizontalAlignment = 'right';

app.Band4DropDownLabel.Position = [89 239 44 15];

app.Band4DropDownLabel.Text = 'Band 4';

% Create Band4DropDown

app.Band4DropDown = uidropdown(app.UIFigure);

app.Band4DropDown.Items = {'Black', 'Brown', 'Red', 'Orange', 'Yellow', 'Green', 'Blue', 'Gold', 'Silver'};

app.Band4DropDown.ItemsData = {'1', '10', '100', '1000', '10000', '100000', '1000000', '0.1', '0.01'};

app.Band4DropDown.Position = [148 235 100 22];

app.Band4DropDown.Value = '1';

% Create Band5DropDownLabel

app.Band5DropDownLabel = uilabel(app.UIFigure);

app.Band5DropDownLabel.HorizontalAlignment = 'right';

app.Band5DropDownLabel.Position = [89 192 44 15];

app.Band5DropDownLabel.Text = 'Band 5';

% Create Band5DropDown

app.Band5DropDown = uidropdown(app.UIFigure);

app.Band5DropDown.Items = {'Brown', 'Red', 'Gold', 'Silver'};

app.Band5DropDown.ItemsData = {'1', '2', '5', '10'};

app.Band5DropDown.Position = [148 188 100 22];

app.Band5DropDown.Value = '1';

% Create ResistorColourCodeDecoderLabel

app.ResistorColourCodeDecoderLabel = uilabel(app.UIFigure);

app.ResistorColourCodeDecoderLabel.FontName = 'Arial Black';

app.ResistorColourCodeDecoderLabel.FontSize = 22;

app.ResistorColourCodeDecoderLabel.FontWeight = 'bold';

app.ResistorColourCodeDecoderLabel.Position = [151 427 373 36];

app.ResistorColourCodeDecoderLabel.Text = 'Resistor Colour Code Decoder';

% Create CalculateButton

app.CalculateButton = uibutton(app.UIFigure, 'push');

app.CalculateButton.ButtonPushedFcn = createCallbackFcn(app, @CalculateButtonPushed, true);

app.CalculateButton.Position = [119 144 100 22];

app.CalculateButton.Text = 'Calculate';

% Create NOTEIfresistoronlyhas4bandsleaveBand3asEmptyLabel

app.NOTEIfresistoronlyhas4bandsleaveBand3asEmptyLabel = uilabel(app.UIFigure);

app.NOTEIfresistoronlyhas4bandsleaveBand3asEmptyLabel.Position = [30 94 336 30];

app.NOTEIfresistoronlyhas4bandsleaveBand3asEmptyLabel.Text = 'NOTE: If resistor only has 4 bands leave Band 3 as ''Empty''';

% Create EditField

app.EditField = uieditfield(app.UIFigure, 'text');

app.EditField.FontColor = [0.902 0.902 0.902];

app.EditField.BackgroundColor = [0.9412 0.9412 0.9412];

app.EditField.Position = [393 128 21 85];

% Create EditField\_2

app.EditField\_2 = uieditfield(app.UIFigure, 'text');

app.EditField\_2.BackgroundColor = [0.9412 0.9412 0.9412];

app.EditField\_2.Position = [427 129 21 84];

% Create EditField\_3

app.EditField\_3 = uieditfield(app.UIFigure, 'text');

app.EditField\_3.BackgroundColor = [0.9412 0.9412 0.9412];

app.EditField\_3.Position = [461 129 22 85];

% Create EditField\_4

app.EditField\_4 = uieditfield(app.UIFigure, 'text');

app.EditField\_4.BackgroundColor = [0.9412 0.9412 0.9412];

app.EditField\_4.Position = [495 129 22 84];

% Create EditField\_5

app.EditField\_5 = uieditfield(app.UIFigure, 'text');

app.EditField\_5.BackgroundColor = [0.9412 0.9412 0.9412];

app.EditField\_5.Position = [545 130 21 83];

% Create HelpButton

app.HelpButton = uibutton(app.UIFigure, 'push');

app.HelpButton.ButtonPushedFcn = createCallbackFcn(app, @HelpButtonPushed, true);

app.HelpButton.Position = [259 15 113 33];

app.HelpButton.Text = 'Help';

% Create ResultsLabel

app.ResultsLabel = uilabel(app.UIFigure);

app.ResultsLabel.FontWeight = 'bold';

app.ResultsLabel.Position = [468 370 49 15];

app.ResultsLabel.Text = 'Results';

% Create InputLabel

app.InputLabel = uilabel(app.UIFigure);

app.InputLabel.FontWeight = 'bold';

app.InputLabel.Position = [151 402 35 15];

app.InputLabel.Text = 'Input';

% Create ResistorIllustrationLabel

app.ResistorIllustrationLabel = uilabel(app.UIFigure);

app.ResistorIllustrationLabel.FontWeight = 'bold';

app.ResistorIllustrationLabel.Position = [435 234 118 15];

app.ResistorIllustrationLabel.Text = 'Resistor Illustration';

% Create ForhelppleaseclickbelowLabel

app.ForhelppleaseclickbelowLabel = uilabel(app.UIFigure);

app.ForhelppleaseclickbelowLabel.FontWeight = 'bold';

app.ForhelppleaseclickbelowLabel.FontAngle = 'italic';

app.ForhelppleaseclickbelowLabel.Position = [241 56 161 15];

app.ForhelppleaseclickbelowLabel.Text = 'For help please click below';

% Create ResultText

app.ResultText = uitextarea(app.UIFigure);

app.ResultText.Editable = 'off';

app.ResultText.HorizontalAlignment = 'center';

app.ResultText.Position = [368 274 253 82];

end

end

methods (Access = public)

% Construct app

function app = ResistorColourDecoderApp

% Create and configure components

createComponents(app)

% Register the app with App Designer

registerApp(app, app.UIFigure)

if nargout == 0

clear app

end

end

% Code that executes before app deletion

function delete(app)

% Delete UIFigure when app is deleted

delete(app.UIFigure)

end

end

end

**Painter Function**

function y = Painter(x,z)

if strcmp(z,'digits')

switch x

case '0'

y = 'black';%

case '1'

y = [0.5 0 0]; %brown

case '2'

y = 'red';%

case '3'

y = [1 0.5 0]; %orange

case '4'

y = 'yellow';%

case '5'

y = 'green';%

case '6'

y = 'blue';%

case '7'

y = 'magenta';%

case '8'

y = [0.7 0.7 0.7]; %grey

case '9'

y = 'white';%

end

elseif strcmp(z,'multiplier')

switch x

case '1'

y = 'black';%

case '10'

y = [0.5 0 0]; %brown

case '100'

y = 'red';%

case '1000'

y = [1 0.5 0]; %orange

case '10000'

y = 'yellow';%

case '100000'

y = 'green';%

case '1000000'

y = 'blue';%

case '0.1'

y = [0.9 0.75 0.35];%gold

case '0.01'

y = [0.8 0.8 0.85]; %silver

end

elseif strcmp(z,'tolerance')

switch x

case '1'

y = '[0.5 0 0]';%brown

case '2'

y = 'red';

case '5'

y = [0.9 0.75 0.35];%gold

case '10'

y = [0.8 0.8 0.85]; %silver

end

end

**Resistor Function Test**

clc;

clear;

num = xlsread("ResistorFunctionTestSheet.xlsx");

k = 1;

correct = 0;

total = 0;

while(num(k,1) ~= 888)

if (num(k,3) == 888)

[r,t,rmin,rmax,rstring] = ResistorFunction(num(k,1),num(k,2),num(k,4),num(k,5));

if (rstring == 'K')

r = r\*1000;

elseif (rstring == 'M')

r = r\*1000000;

end

if ((r == num(k,6)) && (t == num(k,7)))

correct = correct + 1;

end

end

if (num(k,3) ~= 888)

[r,t,rmin,rmax,rstring] = ResistorFunction(num(k,1),num(k,2),num(k,3),num(k,4),num(k,5));

if (rstring == 'K')

r = r\*1000;

elseif (rstring == 'M')

r = r\*1000000;

end

if ((r == num(k,6)) && (t == num(k,7)))

correct = correct + 1;

end

end

total = total + 1;

k = k + 1;

end

fprintf('Resistor function correctly tested %g out of %g inputs\n',correct,total);

**Help App**

classdef Help < matlab.apps.AppBase

% Properties that correspond to app components

properties (Access = public)

UIFigure matlab.ui.Figure

TextArea matlab.ui.control.TextArea

IUnderstandButton matlab.ui.control.Button

MoreInfoButton matlab.ui.control.Button

end

methods (Access = private)

% Button pushed function: MoreInfoButton

function MoreInfoButtonPushed(app, event)

web("http://www.physics-and-radio-electronics.com/electronic-devices-and-circuits/passive-components/resistors/resistorcolorcode.html")

end

% Button pushed function: IUnderstandButton

function IUnderstandButtonPushed(app, event)

delete(app)

end

end

% App initialization and construction

methods (Access = private)

% Create UIFigure and components

function createComponents(app)

% Create UIFigure

app.UIFigure = uifigure;

app.UIFigure.Position = [100 100 395 246];

app.UIFigure.Name = 'UI Figure';

% Create TextArea

app.TextArea = uitextarea(app.UIFigure);

app.TextArea.Position = [16 62 365 168];

app.TextArea.Value = {'A resistor consists of 4 or 5 colour coded bands to represent values that can be used to calculate the resistance and tolerance. To read these values the resistor must be orientated with the tolerance band to the right. This band can be brown, red, gold or silver and there is a slightly larger gap between this band and the other 3-4 bands. It will also appear slightly thicker than the other bands. Once correctly orientated with the identified tolerance band on the right hand side, simply input the bands from left to right (furthermost left band is band 1) and press calculate. Make sure to leave band 3 empty if the resistor being read is a 4 band resistor.'};

% Create IUnderstandButton

app.IUnderstandButton = uibutton(app.UIFigure, 'push');

app.IUnderstandButton.ButtonPushedFcn = createCallbackFcn(app, @IUnderstandButtonPushed, true);

app.IUnderstandButton.Position = [59 19 100 22];

app.IUnderstandButton.Text = 'I Understand';

% Create MoreInfoButton

app.MoreInfoButton = uibutton(app.UIFigure, 'push');

app.MoreInfoButton.ButtonPushedFcn = createCallbackFcn(app, @MoreInfoButtonPushed, true);

app.MoreInfoButton.Position = [228 19 100 22];

app.MoreInfoButton.Text = 'More Info';

end

end

methods (Access = public)

% Construct app

function app = Help

% Create and configure components

createComponents(app)

% Register the app with App Designer

registerApp(app, app.UIFigure)

if nargout == 0

clear app

end

end

% Code that executes before app deletion

function delete(app)

% Delete UIFigure when app is deleted

delete(app.UIFigure)

end

end

end