**Name:** Donark Patel (DP663)

**Class:** IS114-451

**Professor:** Maura Deek

**Date:** 11/1/2019

**1. Formulating the Problem**

**1.1 Problem Description**

Create a program that print out palindrome words using recursion.

**1.2 Verbalization**

*What is the goal?*

Create a GUI for users to enter characters. When the user inputs the characters it print out palindrome words.

*What are the givens?*

User’s inputs the characters. Maximum 5 characters are allowed.

*What are the unknowns?*

List of palindromes words.

1.3 **Information Elicitation**

*Goal*: Create a GUI for users to enter characters. From the characters are inputted print out palindrome words.

*Givens*: User’s inputs the characters. Maximum 5 characters are allowed.

*Unknowns*: List of palindromes words.

*Conditions*:

Maximum 5 characters are allowed to be entered by user.

Outputted words should be palindromes words

**2. Planning the Solution**

**2.1 Solution Strategy**

Ask the user to input the characters. Validated the input. Once the characters are validated. Each character is inserted in the char array. Char array, length of the array and index of the array is sent to the recursion method. Recursion method prints out the combination of palindromes words and make a call itself.

**)2.2 Goal Decomposition**

*Sub-goal 1*: Create the GUI panel.

*Sub-goal 2*: Validate the input.

*Sub-goal 3*: Create array of inputted characters.

*Sub-goal 4*: Call the recursion method.

*Sub-goal 5*: Print out the output.

**2.3 Resources**

*Relevant formulas*

int r = (input.length() -1); //Size of a combination to be printed

**2.4 Data Organization and Description**

Input:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Origin | Used in Sub-goal # |
| frame | GUi Frame | User | 1 |
| lblNewLabel | Enter Alphabets | User | 1, 2 |
| textField | TextBox for Characters | User | 1, 2 |
| btnSubmit | Submit Button | User | 1, 2 |
| btnClear | Clear Button | User | 1 |
| btnExit | Exit Button | User | 1 |
| input | TextBox input | User | 1, 2, 3, 4 |
| inputLength | Length of inputted string | User | 1, 2, 3, 4 |
| charArray | Array to store char | User | 1, 3, 4 |

Output:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Origin | Used in Sub-goal # |
| Size | Size of the string output | Screen | 5 |
| textArea | TextArea to Output String | Screen | 5 |

**3. Designing the Solution**

**)3.1 Structure Chart**

*First Level Decomposition*

*Goal Refinement*

**Sub-goal 1**

Create the GUI Panel

**Sub-Goal 1.1**

From main class initialize the GUI panel class

**Sub-goal 2**

Validate the input

**Sub-goal 2.1**

Check the length of the input using .length() method.

**Sub-goal 2.2**

If the length is less, then 1 or grater then 5. Then output error.

**Sub-goal 3**

Create array of inputted characters.

**Sub-goal 3.1**

Using loop add each character of the string into an array.

**Sub-goal 4**

Call the recursion method.

**Sub-goal 4.1**

Print the combination

**Sub-goal 4.2**

Make a recursion call.

**Sub-goal 5**

Print out the output.

**Sub-goal 5.1**

Output the combination string in the textArea.

*Second Level Decomposition*

**()3.2 Module and Data Specifications**

**Name**: initialize – handles the frame elements.

**Input**: frame, lblNewLabel, textField, scrollPane, textArea, scrollBar, btnSubmit, btnClear, btnExit.

**Output**: None

**Logic**: None

**Name**: SubmitButtonListner – Submit button ActionListener

**Input**: String

**Output**: String

**Logic**: Using the lop structure. Inert each charger of the inputted string into the char Array.

**Name**: Main – Class Frame class.

**Input**: None

**Output**: GUI panel

**Name**: combination – Create the combination of each index in the array, Print the combination.

**Input**: char[] charArray, char[] data, int start, int end, int index, int r

**Output**: TextArea in GUI panel

**Name**: printCombinationData – Create a temporary char Array

**Input**: char array, length of the output string, length on the input string.

**Output**: None

**Data:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Structure |
| frame | JFrame | Object |
| lblNewLabel | JLabel | Object |
| textField | JTextField | Object |
| scrollPane | JScrollPane | Object |
| textArea | JTextArea | Object |
| btnSubmit | JButton | Object |
| btnClear | JButton | Object |
| btnExit | JButton | Object |
| Input | String | Variable |
| OutputLength | Int | Variable |
| inputLength | Int | Variable |
| charArray | Char Array | Array |
| input2 | String | Variable |
| Input3 | String | Variable |

**3.3 Algorithm**

*Logic*

1.0: Create the GUI Panel

1.1: From main class initialize the GUI panel class.

2.0: Validate the input

2.1: Check the length of the input using .length() method.

2.2: If the length is less, then 1 or grater then 5. Then output error.

3.0: Create array of inputted characters.

3.1 Using loop add each character of the string into an array.

4.0: Call the recursion method.

4.1 Print the combination.

4.2 Make a recursion call.

5.0: Print out the output.

5.1: Output the combination string in the textArea.

*Algorithm Description*

The program displays the GUI, where user can input the string. Inputted string is validated. String length cannot be less than 1 and greater than 5 characters. Inputted string is pass into a loop. In the loop, each character of the string is inserted into the Char array. Char array, Int array, start index, end index and the size of the output passed into a recursion method called combination. In the recursion method it prints out the first combination. Then a recursion call is initiated with start index plus one and index plus one. All output is printed in textArea.

**4. Translation**

**4.1** **Source Code**

**import** java.awt.EventQueue;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JOptionPane;

**import** java.awt.Font;

**import** javax.swing.JTextField;

**import** javax.swing.JTextArea;

**import** javax.swing.JScrollBar;

**import** javax.swing.JButton;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** javax.swing.JScrollPane;

**public** **class** Frame1 {

**private** JFrame frame; //GUI Frame

**private** JTextField textField; //Input Textbox

**private** JTextArea textArea; //Output TextArea

**private** JScrollPane scrollPane; //Panel

/\*\*

\* Launch the application.

\*/

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

EventQueue.*invokeLater*(**new** Runnable() {

**public** **void** run() {

**try** {

Frame1 window = **new** Frame1();

window.frame.setVisible(**true**);

} **catch** (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the application.

\*/

**public** Frame1() {

initialize();

}

/\*\*

\* Initialize the contents of the frame.

\*/

**private** **void** initialize() {

// Setting up the frame

frame = **new** JFrame();

frame.setBounds(100, 100, 549, 299);

frame.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

frame.getContentPane().setLayout(**null**);

//Label for Alphabets

JLabel lblNewLabel = **new** JLabel("Enter Alphabets:");

lblNewLabel.setFont(**new** Font("Tahoma", Font.***PLAIN***, 16));

lblNewLabel.setBounds(45, 44, 118, 20);

frame.getContentPane().add(lblNewLabel);

//textbox for inpput

textField = **new** JTextField();

textField.setBounds(173, 46, 86, 20);

frame.getContentPane().add(textField);

textField.setColumns(5);

//ScrollPane for text area.

scrollPane = **new** JScrollPane();

scrollPane.setBounds(41, 85, 353, 152);

frame.getContentPane().add(scrollPane);

//textArea for output

textArea = **new** JTextArea();

textArea.setFont(**new** Font("Monospaced", Font.***PLAIN***, 18));

scrollPane.setViewportView(textArea);

//ScrollPane for text area.

JScrollBar scrollBar = **new** JScrollBar();

scrollBar.setBounds(377, 85, 17, 152);

frame.getContentPane().add(scrollBar);

//Button for submit to input

JButton btnSubmit = **new** JButton("Submit");

btnSubmit.setBounds(275, 45, 89, 23);

frame.getContentPane().add(btnSubmit);

//Button for clearing the data

JButton btnClear = **new** JButton("Clear");

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

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

textArea.setText("");

textField.setText("");

}

});

btnClear.setBounds(421, 45, 89, 23);

frame.getContentPane().add(btnClear);

//Button for exit

JButton btnExit = **new** JButton("Exit");

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

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

System.*exit*(0);

}

});

btnExit.setBounds(421, 99, 89, 23);

frame.getContentPane().add(btnExit);

btnSubmit.addActionListener(**new** SubmitButtonListner() {

});

}

**public** **class** SubmitButtonListner **implements** ActionListener {

@Override

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

// **TODO** Auto-generated method stub

String input = textField.getText(); //Inputed string

**if** (input.length() > 5 || input.length() <= 0) // Validating input

{

JOptionPane.*showMessageDialog*(**null**,"Character length should be \n > 0 \n and \n <= 5");

**return**;

}

input += textField.getText();

**int** OutputLength = (input.length() -1); //Size of a combination to be printed

**int** inputLength = input.length(); //length of the input.

**char** [] charArray = **new** **char** [inputLength]; //Char array

**for**(**int** i = 0; i < inputLength; i++)

{

charArray[i] = input.charAt(i);

}

**do** {

printCombinationData(charArray,inputLength, OutputLength );

OutputLength--;

}**while**(OutputLength>=1);

}

**private** **void** printCombinationData(**char**[] charArray, **int** inputLength, **int** OutputLength) {

**char** data [] = **new** **char**[OutputLength];

combination(charArray, data, 0, inputLength-1, 0, OutputLength);

}

//recursion method.

**private** **void** combination(**char**[] charArray, **char**[] data, **int** start, **int** end, **int** index, **int** OutputLength)

{

String input = "";

**if** (index == OutputLength)

{

**if** (OutputLength == 2 || OutputLength == 4 || OutputLength == 6 || OutputLength == 8 || OutputLength == 10) {

**for** (**int** j=0; j<OutputLength; j++)

input = input + data[j];

StringBuilder input1 = **new** StringBuilder();

input1.append(input);

input1 = input1.reverse();

String input2 = input1.substring(0);

String input3 = input + input2;

**if**(input3.length() <= (charArray.length))

textArea.append(input3 + "\n");

**return**; }

**else** {

**for** (**int** j=0; j<OutputLength; j++)

input = input + data[j];

StringBuilder input1 = **new** StringBuilder();

input1.append(input);

input1 = input1.reverse();

String input2 = input1.substring(1);

String input3 = input + input2;

**if**(input3.length() <= charArray.length)

textArea.append(input3 + "\n");

**return**;

}

}

**for** (**int** i=start; i<=end && end-i+1 >= OutputLength-index; i++)

{

data[index] = charArray[i];

combination(charArray, data, i+1, end, index+1, OutputLength);

}

}

}

}

**4.2 Program and Module Description**

Frame1

Build the GUI frame.

SubmitButtonListner

Handle the submit button action.

actionPerformed

Submit button actions.

printCombinationData

Create a temporary char Array

Combination

Create the combination of each index in the array, Print the combination.

Main

Initialize the frame.

**5. Solution Testing**

Test Case:

|  |  |  |  |
| --- | --- | --- | --- |
| Case | Input | Status of input | Reason |
| 1 |  | Invalid | Cannot be empty |
| 2 | 123 | Invalid | Name Cannot contain numbers |
| 3 | ABCDEF | Invalid | String input length cannot be more then 5 |
| 4 | ABC | Valid |  |

**(This is just a sample what you need to submit are actual screenshots of I/O or files)**







