Santosh Suwal

ID# 0793558

**Recursive Helper Method.**

As the topic recursive helper method is used to make the recursive easier and faster. Slight variation in code can bring variation in its performance speed and make it easy to understand. To use this technique, a clear thought process is necessary that is a programmer has to think clearly what he/she is doing and write it down so that a nice frame for the program can be built.

Palindrome Example:

public boolean isPalindrome()   
{   
int length = text.length();   
// Separate case for shortest strings.   
if (length <= 1) return true;   
// Get first and last characters, converted to   
      lowercase.   
char first = Character.toLowerCase(text.charAt(0));   
char last = Character.toLowerCase(text.charAt(length - 1));   
if (Character.isLetter(first) && Character.isLetter(last))   
{   
   // Both are letters.   
   if (first == last) // Remove both first and last character.  
   {

      Sentence shorter = new Sentence(text.substring(1, length - 1));   
      return shorter.isPalindrome(); //Makes a new string  
   }   
   else   
      return false;   
   }  
   else if (!Character.isLetter(last)) // Remove last character.  
   {   
      Sentence shorter = new Sentence(text.substring(0, length - 1));   
      return shorter.isPalindrome(); //Makes a new string or sentence  
   }   
   else // Remove first character.  
   {

      Sentence shorter = new Sentence(text.substring(1));   
      return shorter.isPalindrome(); // Makes a new string or sentence  
   }   
}

This program runs and checks for a sentence to be palindrome. The program basically checks for the first and last letters and remove it then form another string without that letters. That is this program constructs a sentence or a string every time it runs. The highlighted codes passes a new sentence and is a recursive method.

Instead of constructing sentence for every recursion process we can use a recursive helper method to select a substring and form a recursive statement.

public boolean isPalindromeHelper(int start, int end) {

// separate case for substrings of length 0

// and 1

if( start >= end )return true;

// Get first and last characters, converted to

// lowercase

char first = Character.toLowerCase(text.charAt(start));

char last = Character toLowerCase(text charAt(end));

if( Character.isLetter(first) && Character.isLetter(last))

{

if(first == last)

{

// Test substring that doesn’t contain the matching

// letters

return isPalindromeHelper(start + 1, end – 1);

}

else return false;

}

else if (!Character.isLetter(last))

{

return isPalindromeHelper(start, end – 1);

}

else

{

// Test substring that doesn’t contain the first

// character

return isPalindromeHelper(start + 1, end);

}

}

Now, this program needs a method to keep it going on, which is termed as recursive helper method.

public boolean isPalindrome()

{

return isPalindromeHelper(0, text.length()-1);

}

Here, the recursive method is isPalindrome(). This method chain of recursive action by calling isPalindromeHelper(0, text.length()-1). This way it is easier to understand what is happening in the program.