StringADT

This project is to study the built in classes String, StringBuffer and StringTokenizer. We will going to explore all the important functions of these classes by creating our own ADT with defined operations allowed for that ADT, finally we will develop an console based Hangman Game.

Project Name: StringADT

Package used:

package 1: com.project1.interface – All base interfaces are defined here.

package 2: com.project1.hangman - all the classes related to hangman application are

defined here

package 3:com.project1.mystring – all the classes related to mystring class defined here

package4:com.project1.lexical – all the classes related to lexical class is defined here

package 1: com.project1.interface

Description of the interfaces:

Interface name	**Description**
`StringBase`	Interface for my string class
Method name	***descrition***

public abstract void `showStructure(String str);`	`/*`
	`showStructure method is used to print input String str in th below pattern`
	`0 1 2 3 4 5`
	`H e l l o`
	`The characters in a String can be manipulated one at a time with the use of the charAt(int index) method.*/`

public boolean `lessThan(String leftstring, String rigthstring);`	`/*` `This method use the String class method compareTo to determine if the leftString is less than the rightString.`

	*/
public boolean gtrThan(String leftstring, String rigthstring);	/* This method use the String class method compareTo to determine if the leftString is greater than the rightString. */
public abstract String findSubstring(String teststr1, **int** start, **int** count);	/* This method uses substring and length and returns a substring of testStr1 starting at position start and extracting count characters if that many characters exist in the String testStr1. */
void firstLtrWord(String inputstring);	/* this method a combination of length, charAt, substring, and indexOf can be used to find the first letter, the first word, and the last letter in a given sentence or phrase. This method prints the following lines of information. String is: The string's length is:

	The first letter is: *The last letter is:* *The first word is:* *If the String is empty, this method prints* *String is:* *The string's length is: 0* *The string is empty! No more data to print.* */*
public int strCharCount(String inputstring, **char** ch);	*/** *This method uses the String method indexOf to return a count of the number of times the character* *ch occurs in the String inputString* */*

Interface name	**Description**
StringTokenizerInterface	Interface for lexical class
Method name	***descrition***
public abstract String firstToken(String inputStr);	*/** *this method will return the first token from the*

	input string *by using the nextToken()* *of StringTokenizer* **/*
int tokenCount(String inputStr);	*/** *this method will return* *the total no of tokens* *present in* *inputString by using the* *countTokens() of* *StringTokenizer* **/*
void showAllToken(String inputStr);	*/** *this method will going to* *print all the available* *tokens in* *inputstring by using the* *hasMoreToken() method and* *nextToken()* *method of StringTokenizer* **/*

package 3:com.project1.mystring

Description of the class:

class	**Description**
MyString	Implements StringBase interface
Method name	***descrition***
Override all the methods of interface	Use the method from built in String class to implement all the method.
Main method	To test our implementation by calling all the overridden methods

package4:com.project1.lexical

Description of the class:

class	**Description**
Lexical	Implements StringBase interface This class will print any entered input strings into tokens. Tokens are created by separating each word from sentence. By using StringTokenizer class.
Method name	***descrition***
Override all the methods of interface	Use the method from built in String class to implement all the method.
Main method	To test our implementation by calling all the overridden methods

package 2: com.project1.hangman

Description of the class:

class	**Description**
Hnagman	This class contains all the functionality for implementing hangman game you will use String and StringBuffer objects to create a two-person version of the ÒHangmanÓ word-guessing game. The game begins with one player entering a secret word that is scrolled off the screen before the other player sits down to play. A blank guess template then appears on the screen. This template is the same length as the secret word but has dashes in place of the letters in the word. The player attempting to guess the secret word enters letters one at a time. After each guess, the guess template is updated (if necessary) to show which letters in the secret word match the letter guessed. For example, if the secret word is ÒscissorsÓ, guessing ÔsÕ as the Þrst correctly guessed letter results in the following changes in the guess template: Guess a letter: s s--ss--s This process continues until the guess template matches the secret word. The number of guesses is then output. A sample game is shown below. Enter the secret word: test (This scrolls off the screen) ---- Guess a letter: a ---- Guess a letter: e -e-- Guess a letter: n -e-- Guess a letter: s -es- Guess a letter: t

	test=test You guessed the word in 5 guesses. There are three key methods in this Hangman program.
Method name	***descrition***
`public static int startgame(StringBuffer word)`	Use this method start the game by sending the secret word and this method will return the total number of guesses. This method uses BufferReader to read input. This method will call printword() to print template and updated template.
`public static void printword(StringBuffer guess)`	This method is used to print the guess template
`public static int getIndexFromWord(char letter,StringBuffer word,int index)`	This method will return the index of letter from string buffer word, and search for index from given index
Main method	To test our implementation by calling all the overridden methods