String Cheat Sheet

**Palindromes**

**public** **class** Palindrome

{

**public** **static** **boolean** isPalindrome(String stringToTest)

{

String workingCopy = *removeJunk*(stringToTest);

String reversedCopy = *reverse*(workingCopy);

**return** reversedCopy.equalsIgnoreCase(workingCopy);

}

**protected** **static** String removeJunk(String string)

{

**int** i, len = string.length();

StringBuffer dest = **new** StringBuffer(len);

**char** c;

**for** (i = (len - 1); i >= 0; i--) {

c = string.charAt(i);

**if** (Character.*isLetterOrDigit*(c))

{

dest.append(c);

}

}

**return** dest.toString();

}

**protected** **static** String reverse(String string)

{

StringBuffer sb = **new** StringBuffer(string);

**return** sb.reverse().toString();

}

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

{

String string = "Madam, I'm Adam.";

**if** (*isPalindrome*(string)) {

System.*out*.println("It IS a palindrome!");

}

}

}

**String Matchings**

**private** **static** HashMap<Character, Integer> *shiftTable* = **new**

HashMap<Character, Integer>();

**private** **static** **int** *counter* = 0;

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

{

Scanner scan = **new** Scanner(System.*in*);

String string = scan.nextLine();

*makeAShiftTable*(string);

String text = scan.nextLine();

**int** answer = *findMatch*(text, string);

System.*out*.println("Match found for " + string + " at " + answer + "

(if starting at index 0)." + "\n in " + text + "\n in ");

System.*out*.println("Number of comparisons: " + *counter*);

}

**private** **static** **int** findMatch(String text, String string)

{

//the position at the end of the string

**int** i = string.length() - 1;

//search for a match

//the starting point for searching through the text

**int** k = string.length()-1;

**while**(k <= text.length() - 1 && i >= 0)

{

//if end of string char is not equal to the current

//text's char

//call the shiftTable value on it

**if**(string.charAt(i) != text.charAt(k))

{

**if**(k - string.length() <= 0)

{

k += string.length();

}

//if the char is not in the table go the length of

//the string

**else** **if**(*shiftTable*.get(text.charAt(k)) != **null**)

{

**if**(i != string.length())

{

i = string.length()-1;

}

//i += shiftTable.get(string.charAt(i));

k += *shiftTable*.get(text.charAt(k)) - 1;

}

**else**

{

**if**(i != string.length())

{

i = string.length()-1;

}

//length of the string

k += string.length() + 1;

}

}

//else if it does equal look at the character to the left

//and compare

**else**

{

*counter*++;

//m = k;

//if the whole string has been looked at (i @ 0)

//return the

//position of k (where it is in the text).

**if**( i == 0)

{

**return** k; // - string.length();

}

**else** //if(k != 0)

{

i--;

k--;

}

}

}//if the match isn't found return -1

**return** -1;

}

/\*\*

\* ABCDE

\* **@param** patternGiven

\*/

**public** **static** **void** makeAShiftTable(String patternGiven)

{

**for**(**int** i = 0; i < patternGiven.length(); i++)

{

**char** letter = patternGiven.charAt(i);

**int** shiftAmt = patternGiven.length() - patternGiven.lastIndexOf(letter);

*shiftTable*.put(letter, shiftAmt);

}

}