Problem 1: Palindromes

Your program is to read a single line of user input and tell if yes or no it is a palindrome. If user inputs "quit" then exit the program.

A palindrome is a string that read from left to right or right to left looks the same.

For instance "abcba" is identical whether we read it from left to right or from right to left. Your program is expected to be case-sensitive.

On top of this definition we will consider the following pairs of characters to be reversible:

( ), [ ], { }, < >

This means that when reading from left to right we encounter the character '(' we should encounter instead the character ')' when reading from right to left.

For instance the string "ab(c)ba" will be considered to be a palindrome while "ab(c(ba" will not.

Here are some examples of palindromes: aabbcbbaa a

aa

a{}a ( ) (a) ver{(aga)}rev

and some examples of non-palindromes: ab

( a(b(a

a(a a{b>c

Example of what your program output should look like:

>> input phrase (type 'quit' to exit the program):

aabbcbbaa
This is not a palindrome.

>> input phrase (type 'quit' to exit the program):
(a)

This is a palindrome.

>> input phrase (type 'quit' to exit the program):
quit

**Solution**

/** 2008 WCU Computer Science High School Programming Contest
 * Solution to Problem One, Copyright 2008, all rights reserved
 * Western Carolina University
 * Department of Mathematics and Computer Science */
import java.io.*;
import java.util.*;

public class Problem1
{
    /** This class returns the 'reverse' version of a character is
     * such exists; otherwise the character itself is returned. */
    public static char reverseChar(char c)
    {
        switch(c) {
            case '(': return ')';
            case ')': return '(';
            case '{': return '}';
            case '}': return '{';
            case '[': return ']';
            case ']': return '[';
            case '<': return '>';.
            case '>': return '<';
            default: return c;
        }
    }
}
/** This function returns true or false whether the passed * argument is a palindrome or not. */
public static boolean isPalindrome(String str)
{
    // try matching char-by-char reading from both sides
    // at once
    for(int i = 0; i < str.length() / 2; i++)
    {
        if (str.charAt(i) != reverseChar(
            str.charAt(str.length()-1-i)))
        {
            return false;
        }
    }

    // special case - strings of odd length:
    // the central character cannot be a reversible character
    if (str.length() % 2 == 1)
    {
        if (str.charAt(str.length() / 2) != reverseChar(str.charAt(str.length() / 2)))
            return false;
    }

    return true;
}

public static void main(String[] args)
{
    Scanner scanIn = new Scanner(System.in);

    while(true)
    {
        // get user input
        System.out.println(">> input phrase (type 'quit' to exit " +
            "the program):");
        String line = scanIn.nextLine();

        // check for need to quit
        if (line.equals("quit")) System.exit(0);

        // check for palindrome
        if (isPalindrome(line))
            System.out.println("\nThis is a palindrome.\n");
        else
            System.out.println("\nThis is not a palindrome.\n");
    }
    // end of while
}
Problem 2: Passwords

Your program will read user input and return a numeric value associated with that input.
If user inputs "quit" then exit the program.

The user input is to be interpreted as a potential password.

The numeric value that you will return will be computed using the set of rules below.
That value indicates the 'strength' of the password. The bigger the number, the more
secure the password is considered to be.

Rules to compute the strength of a password:

0) By default the strength is 0. Use the rules below to find the proper value
   associated with your password.

1) If there are at least 8 characters, add +10 to the strength.
2) If longer than 8 characters, add +1 per every two-characters.
3) If the password contains both upper and lower case, add +1 to the strength.
4) If the password contains at least one numeric character, add +1 to the strength.
5) If the password contains at least one special character, add +1 to the strength.
6) For every two consecutive characters that aren't of the same type (alphabetic,
   numeric, special), add +1 to the strength.

Note:

* Alphabetic characters can be either lowercase (a..z) or uppercase (A..Z).
* Special characters are neither alphabetic or numeric.

Example of what your program output should look like:

```
>> password (type 'quit' to exit the program): abc
strength = 0

>> password (type 'quit' to exit the program): abcdefgh
strength = 10

>> password (type 'quit' to exit the program): abcdefghi
strength = 10

>> password (type 'quit' to exit the program): abcdefghij
strength = 11

>> password (type 'quit' to exit the program): abcdefghij0
strength = 13
```
Solution

```java
import java.io.*;
import java.util.*;

public class Problem3 {

    /** This function helps popping values out of a stack. */
    public static String pop(String stack, char c) {
        if (stack.length() == 0) return null;

        if (stack.charAt(stack.length() - 1) == c) {
            return stack.substring(0, stack.length() - 1);
        } else {
            return null;
        }
    }

    /** This function computes the strength of a password passed as argument. */
    public static boolean check(String str) {
        String stack = "";

        for (int i = 0; i < str.length(); i++) {
            switch (str.charAt(i)) {
                case '{': case '(': case '[':
                    stack = stack + str.charAt(i);
                    break;
                case '}':
                    stack = Problem3.pop(stack, '{');
                    break;
            }
        }

        return stack.length() == 0;
    }

    public static void main(String[] args) {
        String password = "A{bc}defghij0";
        System.out.println(check(password));

        password = "0123456789";
        System.out.println(check(password));

        password = "66Nk+79=abc";
        System.out.println(check(password));

        password = "quit";
        System.out.println(check(password));
    }
}
```
if (stack == null) return false;
break;
case ')':
    stack = Problem3.pop(stack,'(');
    if (stack == null) return false;
    break;
case ']':
    stack = Problem3.pop(stack,'[');
    if (stack == null) return false;
    break;
}
return stack.length() == 0;
}

public static void main(String[] args)
{
    Scanner scanIn = new Scanner(System.in);
    while (true)
    {
        // get user input
        System.out.print("\> user input (type 'quit' to " +
                "exit the program): ");
        String line = scanIn.nextLine();
        // check for need to quit
        if (line.equals("quit")) System.exit(0);
        // check the input validity
        if (Problem3.check(line))
            System.out.println("Valid.
        ");
        else
            System.out.println("Invalid.
        ");
    } // end of while
}
Problem 3: Parentheses Check

Your program will read user input and return a message indicating if the input was valid or not. If user inputs "quit" then exit the program.

The user input is to be interpreted as a single line of text.

You will parse this string of text and indicate whether parentheses, square brackets, and curly braces are properly opened and closed. Any input that is not a parenthesis/bracket/brace can be ignored.

For instance here a few things you may want to consider:

* Is there a closing parenthesis/bracket/brace for every opening one?
* Are the parenthesis/bracket/brace blocks properly embedded one into another?
* anything that is not a parenthesis/bracket/brace can be ignored.

Examples of invalid input:

(is very ) is invalid because it misses a closing parenthesis
(is very)) is invalid because there is an extra closing parenthesis
(aaa [bbb ) ccc]ddd is invalid because it closes the parentheses before the brackets

Example of valid input:

apples (are very [very] good {at} (k(ee)[p]ing ) the [doctor{away}].

Example of what your program output should look like:

>> user input (type 'quit' to exit the program): ((is very )
Invalid.

>> user input (type 'quit' to exit the program): (is very) )
Invalid.

>> user input (type 'quit' to exit the program): ((is very) )
Valid.

>> user input (type 'quit' to exit the program): quit

Solution

/** 2008 WCU Computer Science High School Programming Contest
* Solution to Problem Three, Copyright 2008, all rights reserved

7
import java.io.*;
import java.util.*;

public class Problem3 {

    /** This function helps popping values out of a stack. */
    public static String pop(String stack, char c) {
        if (stack.length() == 0) return null;
        if (stack.charAt(stack.length() - 1) == c) {
            return stack.substring(0, stack.length() - 1);
        } else {
            return null;
        }
    }

    /** This function computes the strength of a password passed as argument. */
    public static boolean check(String str) {
        String stack = "";
        for (int i = 0; i < str.length(); i++) {
            switch (str.charAt(i)) {
                case '{': case '(': case '[': 
                    stack = stack + str.charAt(i);
                    break;
                case '}':
                    stack = Problem3.pop(stack, '{');
                    if (stack == null) return false;
                    break;
                case ')':
                    stack = Problem3.pop(stack, '(');
                    if (stack == null) return false;
                    break;
                case ']':
                    stack = Problem3.pop(stack, '[');
                    if (stack == null) return false;
                    break;
            }
        }
        return stack.length() == 0;
    }

    public static void main(String[] args) {
        Scanner scanIn = new Scanner(System.in);

        while (true) {
            // get user input

            String[] args = scanIn.nextLine().split(" ");
            String str = args[0];
            if (str.equals("end")) {
                return;
            }
            System.out.println("Strength: "+check(str));
        }
    }
}
System.out.print(">> user input (type 'quit' to " +
"exit the program): ");
String line = scanIn.nextLine();

// check for need to quit
if (line.equals("quit")) System.exit(0);

// check the input validity
if (Problem3.check(line))
    System.out.println("\nValid.\n");
else
    System.out.println("\nInvalid.\n");

} // end of while
}
Problem 4: Rock Paper Scissors

Your program will read user input and use it to play a round of the well-known rock/paper/scissors game. If user inputs "quit" then exit the program.

The user input is case insensitive and the user is expected to type "paper", "rock", or "scissors". If anything else is typed then call for another round giving an invalid input error message.

At each turn, your program is to pick arbitrarily and unsystematically one of "paper", "rock", "scissors." You shouldn't use user input in deciding what to play.

After your program is done picking what to play, compare the user input and the computer's choice, and indicate who won that round. If there is a draw say so.

Here is a table describing which choice wins over which other choices:

<table>
<thead>
<tr>
<th></th>
<th>Paper</th>
<th>Rock</th>
<th>Scissors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paper</td>
<td>draw</td>
<td>usr wins</td>
<td>cpu wins</td>
</tr>
<tr>
<td>Rock</td>
<td>cpu wins</td>
<td>draw</td>
<td>usr wins</td>
</tr>
<tr>
<td>Scissors</td>
<td>usr wins</td>
<td>cpu wins</td>
<td>draw</td>
</tr>
</tbody>
</table>

Example of what your program output should look like:

Please type rock, paper, or scissors (type 'quit' to exit the program).

>> SCISSORS

Computer Choice: 2

The computer won (rock > scissors).

Please type rock, paper, or scissors (type 'quit' to exit the program).

>> rock

Computer Choice: 2

This round is a draw.

Please type rock, paper, or scissors (type 'quit' to exit the program).

>> paper

Computer Choice: 2

You won (paper > rock).

Please type rock, paper, or scissors (type 'quit' to exit the program).
Solution

/** 2008 WCU Computer Science High School Programming Contest
 * Solution to Problem Four, Copyright 2008, all rights reserved
 * Western Carolina University
 * Department of Mathematics and Computer Science
 */
import java.io.*;
import java.util.*;

public class Problem4
{
    /** Given the userChoice and the computerChoice this function
     * indicates is the computer won. */
    public static boolean cpuWins(int userChoice, int cpuChoice)
    {
        // 0 --> paper
        // 1 --> scissors
        // 2 --> rock
        return (cpuChoice == 0 && userChoice == 2) ||
            (cpuChoice == 1 && userChoice == 0) ||
            (cpuChoice == 2 && userChoice == 1);
    }

    /** This returns the string associated with a number for
     * paper/rock/scissors. */
    public static String choiceString(int choice)
    {
        switch(choice) {
            case 0: return "paper";
            case 1: return "scissors";
            case 2: return "rock";
            default: return "(invalid)";
        }
    }

    public static int processInput()
    {
        Scanner scanIn = new Scanner(System.in);
        boolean gotInput = false;
        int userChoice = 0;

        while (!gotInput)
        {
            // get user input
            System.out.print("Please type rock, paper, or scissors " +
                 "(type 'quit' to exit the program).\n>> ");
            String line = scanIn.nextLine().toLowerCase();
            userChoice = 0;

            }
// check for need to quit
if (line.equals("quit")) System.exit(0);

// check user input
if (line.equals("paper"))
    { userChoice = 0; gotInput = true; }
else if (line.equals("scissors"))
    { userChoice = 1; gotInput = true; }
else if (line.equals("rock"))
    { userChoice = 2; gotInput = true; }
else
    System.out.println("Invalid input. Try again.\n");
} } 
return userChoice;

public static void main(String[] args)
{ Random rd = new Random( System.currentTimeMillis() );

    while (true)
    {
        int userChoice = Problem4.processInput();

        // the computer picks a random choice
        int computerChoice = Math.abs(rd.nextInt() % 3);
        System.out.println("Computer Choice:" + 
                Problem4.choiceString(computerChoice));

        // decide who won
        if (userChoice == computerChoice)
            System.out.println("This round is a draw.\n");
        else if (cpuWins(userChoice,computerChoice))
            System.out.println("The computer won (" + 
                    Problem4.choiceString(computerChoice) + " > " + 
                    Problem4.choiceString(userChoice) + ").\n");
        else
            System.out.println("You won (" + 
                    Problem4.choiceString(userChoice) + " > " + 
                    Problem4.choiceString(computerChoice) + ").\n");
    } // end of while
}