Palindrome.java

```java
/**
 * This class allows the user test whether or not the input is a palindrome.
 * Palindromes are defined as an alphanumeric string that is read the same
 * forward as it is backwards. Whitespace, punctuation, and case are all
 * ignored.
 *
 * The user can call the program WITHOUT command-line arguments to be
 * taken to an interactive program that takes in a phrase and returns the
 * evaluation. The user can also call the program WITH command-line arguments
 * and the program will print the evaluation without interactively asking for
 * any additional input.
 *
 * @name    Palindrome
 * @author  Ravi S. Ramphal
 * @class   CCSF CS111B
 * @date    2017.06.29
 * @version 1.0
 */

import java.util.Scanner;

class Palindrome
{
    /**
     * This method takes a string and returns it reversed.
     *
     * @param  string The string that is to be reversed
     * @return String The reversed string
     */
    private static String reverse(String string)
    {
        return (new StringBuilder(string)).reverse().toString();
    }

    /**
     * This is a function that was written to join an array of Strings
     * together. The main usage would be to allow the user to call this
     * program from the command line without having to use quotations to
     * encapsulate the input. However, upon further consideration, this
     * was decided to be an anti-pattern. This method is currently
     * unused, but is left here for reference.
     *
     * @param  array  An array of strings that are to be joined together
     * @return String The elements of the array joined together by spaces
     */
    private static String join(String ... array)
    {
        StringBuilder temp = new StringBuilder();

        for(int i = 0; i < array.length; i++)
        {
            temp.append(array[i]);
            if (i != (array.length - 1))
            {
                temp.append(" ");
            }
        }

        return temp.toString();
    }

    /**
     * This method filters a given string to return only alphanumeric
     * characters. Originally, it was done by using ‘Character.isLetterOrDigit‘
     * (left here for reference); however it was refactored to use
     * ‘[string].replaceAll()‘ using a Regular Expression to filter.
     *
     * @param  input  The string that is to be filtered
     * @return String The filtered string
     */
    private static String filter(String input)
    {
        // String str = "";
        //
        // for(char x : input.toCharArray())
        // {
        //     if (Character.isLetterOrDigit(x)) str += c;
        // }
        //
        // return str.toUpperCase;

        return input.toUpperCase().replaceAll("[^A-Z0-9]", "");
    }

    /**
     * This method tests filters the input and tests whether or not it is read
     * the same forwards and backwards.
     *
     * @param  input   The string that is to be tested
     * @return boolean A boolean with whether or not the input is a palindrome
     */
    private static boolean isPalindrome(String input)
    {
        return filter(input).equals(filter(reverse(input)));
    }

    /**
     * This is a helper method that simply displays instructions to the user.
     */
    private static void printUsageInfo()
    {
        System.out.println("\nEnter a phrase to test whether it is a palindrome."
);
        System.out.println("Type 'exit', 'end', or 'stop' to exit program.");
    }

    /**
     * This method prompts the user with a message and returns the input.
     *
     * @param  prompt A string containing the message that prompts the user
     * @return String A string containing the content that the user has input
     */
    private static String getInput(String ... prompt)
    {
        if (prompt.length > 0) System.out.print(prompt[0]);
        return (new Scanner(System.in)).nextLine();
    }

    /**
     * This method returns whether the user has inputted an exit code.
     * These are either: "exit", "end", or "stop".
     *
     * @param  input   The string that is to be tested
     * @return boolean A boolean to reflect if the input is an exit code
     */
    private static boolean isExitCode(String input)
    {
        return (
            input.equalsIgnoreCase("exit") ||
            input.equalsIgnoreCase("end")  ||
            input.equalsIgnoreCase("stop")
        );
    }

    /**
     * This is the die method written to let the user know that the program
```

```java
     * is exiting and to exit the program.
     */
    private static void die()
    {
        System.out.println("\nExiting.");
        System.exit(0);
    }

    /**
     * This method takes the input, tests to see if it is a palindrome, and
     * then outputs the result to the user.
     *
     * @param input The string that is to be evaluated
     */
    private static void evaluateInput(String input)
    {
        String qualifier = (isPalindrome(input)) ? "IS" : "IS NOT";
        System.out.println("’" + input + "’ " + qualifier + " a palindrome.");
    }

    /**
     * This method loops the user through interactively providing input
     * and seeing the response. It also allows the user to exit.
     */
    private static void loopInteraction()
    {
        for(;;)
        {
            String input = getInput("\nPlease input phrase: ");

            if (isExitCode(input)) die();
            else
            {
                evaluateInput(input);
            }
        }
    }

    /**
     * This is the main function of this class.
     *
     * If the user has not passed in command-line arguments,
     * the program will print out information on how it is to be
     * used and then interactively ask the user for the input
     * that she would like to test.
     *
     * If the user has passed in command-line arguments, the program
     * will loop over each argument and evaluate it independently.
     */
    public static void main(String ... args)
    {
        if (args.length == 0)
        {
            printUsageInfo();
            loopInteraction();
        }
        else
        {
            for (String input : args)
            {
                evaluateInput(input);
            }
        }
    }
}
```