```java
package Hangman;

import java.util.ArrayList;



/**
 * This is the HangmanGame "Controller" class that launches the game and takes user input,
 * and by reading the user input of continuing the game or not, this class also allows users
 * to run multiple rounds of word guessing in a single run.
 * Besides, I wrote two versions in one class, because traditional version is
 * obviously the special version of evil version that contains only one word in
 * the game set. Therefore, only the game controller class will control
 * the version that user is playing.
 * @author Kevin Long
 *
 */
public class HangmanGame {

    // The Game class is only responsible for generating available arraylist set for guessing
    ArrayList<String> original = new ArrayList<String>();

    // This variable controls the word length that computer randomly generates for user
    int available;

    // This variable controls the states of gaming continuing or not
    boolean conti = true;

    // Instantiate this class
    public HangmanGame(ArrayList<String> s){
        this.original = s;
    }


    /**
     * This static method loops through the available word list and return the maximum length
of the world in the
     * world list, so that the random method do not generate word length that is too long and
there is not match in
     * the list.
     * @param words
     * @return
     */
    public static int availableLength(ArrayList<String> words) {

        // Set the initial length of longest word as 0 and update the value
        int maxLength = 0;

        // Loop through the word list
        for (String word: words) {
            if (word.length() > maxLength) {

                // Once there is any word that has a length over maxLength, update the value
of the variable
                maxLength = word.length();
            }
        }
        return maxLength;
    }


    /**
     * This method loops through the available word list and return the ArrayList<String> word list where all words are
     * exactly the same length as given int length.
     * @param length
     * @return
     */
    public ArrayList<String> gameSet(int length) {

        // Create a new ArrayList and populate it with words that has exactly the same length as variable length
        ArrayList<String> list = new ArrayList<String>();


        for(String word: this.original) {
            if (word.length() == length) {


                // Use list.add() method
                list.add(word);
            }
        }


        return list;
    }


    public static void main(String[] args) {
        // TODO Auto-generated method stub


        // Initialize the txt that we will use
        System.out.println("Welcome to HangmanGame! ");
        String filename = "words.txt";

        // Create new file reader instance to manipulate the txt file and return the "pure" word list
        HangmanFileReader fr = new HangmanFileReader(filename);

        // Get clean word list with no leading or trailing whitespaces
        ArrayList<String> lines = fr.getCleanContext();
        System.out.println("System have read the file: " + filename);

        // Create a new HangmanParsing object to parse the lines and return the pure word list
        HangmanParsing rq = new HangmanParsing(lines);
        ArrayList<String> s = rq.matchCleaning();

        // Generate the available length
        int len = HangmanGame.availableLength(s);

        // Use the pure word list to create a HangmanGame object
        HangmanGame newGame = new HangmanGame(s);

        // Initialize several variables that we will use in latter parts
        Random rand = new Random();
        String pattern = "[a-z]+";

        // Notice the user when creating the pure word list successfully
        System.out.println("System created a pure guess-able word list.");

        // Initialize the scanner onbject
        Scanner scanner = new Scanner(System.in);


        // When the conti is true, continue the game
        while(newGame.conti) {


            // Intialize the word list length for creating Hangman Object, use rand.nextInt(i) + 1 to get exactly the word length that is used
            int i = rand.nextInt(len) + 1;

            // Create the gameSet for the game
            ArrayList<String> game = newGame.gameSet(i);


            // If the computer generates the length that has no words in it, repeat the process and generate a new GameSet
            while(game.size() == 0) {
                i = rand.nextInt(len) + 1;
                game = newGame.gameSet(i);
            }

            // Setting Game mode, if j is 0, then we are playing traditional hangmang. If j is 1, then we are playing an evil hangman
            int j = rand.nextInt(2);

            // If the game is traditional, select a random words from gameset arrylist and process (Actually, traditional hangman game is a special version of evil hangman game)
            if(j == 0) {
                int inde = rand.nextInt(game.size());
                ArrayList<String> game1 = new ArrayList<String>();
                game1.add(game.get(inde));
                game = game1;
            }

            // Use game(ArrayList<String>) and i(int, represents the length of the word selected) to create a Hangman object
            Hangman newHangman = new Hangman(game, i);

            // Print the Rule of the hangman Game
            System.out.println("Welcome to Hangman Game!");
            System.out.println(" ");
            System.out.println("You'll play against computer who randomly choose a word(or word group of a specific length;");
            System.out.println("You'll immediately know how many characters are in the word(s), but you won't know you're guessing");
            System.out.println("a word or a dynamic word group;(You'll get to know it afterwards, though!)");
            System.out.println("All characters undiscovered are marked as '-'.");
            System.out.println("If you guess the character right, it will automatically appear on a location (or more)");
            System.out.println("Now try youre best and hit all of them as soon as possible!   ");
            System.out.println(" ");
            System.out.println("------------------------------------Instruction-------------------------------------------------------");
            System.out.println("Please input a single lowercase character, or put the character in string beginning position, such as a, ab, ass");
            System.out.println("------------------------------------------------------------------------------------------------------");

            // Print the length out at the start of the game and ask the user to input
            System.out.println("Guess a letter");

            System.out.println(newHangman.print());

            // Record the very first string printed
            //String s1 = newHangman.print();

            // Make a judgement to see if the user has guessed all the characters correctly
            while(!newHangman.guessAll()) {

                // Get user guess
                String guess = scanner.next();

                // Compare the user input with regex pattern and try to get a lowercase character at first position
                boolean isMatch = Pattern.matches(pattern, "" + guess.charAt(0));
                // If user didn't input a lowercae character at the first position of the string, pop up the error message and asks the user to input again
                while(!isMatch) {
                    System.out.println("You ara typing un-recongized guess format, make sure you input lowercase character in the beginning");
                    System.out.println("Guess a letter");
                    guess = scanner.next();
                    isMatch = Pattern.matches(pattern, "" + ""+guess.charAt(0));
                }

                // System.out.println("Game size: "+newHangman.testSet.size());

                // Print if users have guessed a character that was guessed
                newHangman.remindRepeatance(guess.charAt(0));

                // Implement the updateMap method and update the frequency Map
                newHangman.updateMap(guess.charAt(0));


                // Compare the frequency Map and update the gameset if the group is with any position of the word.
                newHangman.updatePrint(guess.charAt(0));

                // Iterate until all characters that has the maximum frequency are replaced in other two map
//              while (!s1.equals(newHangman.print())){
//                  s1 = newHangman.print();
//                  newHangman.updateArray(guess.charAt(0));
//              }

                // Initialize frequency map
                newHangman.reInitialize();

                // Print another guess requirement for users
                System.out.println("Guess a letter");

                // Print the printMap out to assit user guessing more characters
                System.out.println(newHangman.print());


            }

            // Tell User the game version that they are playing
            if (j == 0) {
                System.out.println("You've just finished a traditional version hangman game!");
            } else {
                System.out.println("You've just finihsed an evil version hangman game!");
            }

            // When all positions are been updated as "has guessd", end this round
            System.out.println("You have guessed all words.  This is the end of this around. Input Y if you want to start another round, or it ends.");

            // Asks the user input and look at if user want to keep on or want to stop
            String keep = scanner.next();

            // If user want to keep the game, start the loop again
            if(!keep.equals("Y")) {
                newGame.conti = false;
            }
        }

        // If the user don't want to stard another round, end the game
        scanner.close();
    }

}
```