## Q1: Your task is to write a program: Hangman using TDD.

**Hangman.java**

```
package hangman;

import java.util.HashSet;
import java.util.Random;
import java.util.Set;
import java.util.Scanner;

class Hangman {

    public static void main(String[] args) {
        int leftLimit = 97; // letter 'a'
        int rightLimit = 122; // letter 'z'
        int targetStringLength = 5;//length of the word
        Random random = new Random();
 //generating random word using pre-built java8 method.
    String wordToGuess = random.ints(leftLimit, rightLimit + 1)
      .limit(targetStringLength)
      .collect(StringBuilder::new, StringBuilder::appendCodePoint, StringBuilder::append)
      .toString();
   guessWord(wordToGuess);
}
public static void guessWord(String wordToGuess) {        int chances = 0;
        Set<String> prevGuesses = new HashSet<>();
        Scanner input = new Scanner(System.in);
        int length = wordToGuess.length();
        char[] wordToGuessChars = wordToGuess.toCharArray(); //creates character array of strings
        char[] censor = wordToGuess.toCharArray();
        System.out.println("Your secret word is: "); // prints an array of chars with the same length as
string
        for (int index = 0; index < length; index++) {
            censor[index] = '_';
        }
        while (!String.valueOf(censor).equals(wordToGuess)) {
            //Initialize all variables in loop
            boolean correct = false; //lets the user know if the letter is in the word or not
            boolean repeated = false; //check if user guessed the same letter twice
            for (int a = 0; a < length; a++) {
                System.out.print(censor[a]);//prints the censored secret word
            }
            System.out.println();
            System.out.println("Type your guess: ");
            String currentGuess = input.next().substring(0, 1);
            char currentGuessChar = currentGuess.charAt(0); //gets char data from scanner
            if (prevGuesses.contains(currentGuess)) //checks if user already guessed the letter
previously
            {
                System.out.println("You already guessed this letter! Try again. Your previous guesses
were: ");
                System.out.println(prevGuesses.stream().reduce("", String::concat));
                repeated = true;
```

```
            }
            prevGuesses.add(currentGuess);
            //if the guess is not a duplicated guess, it checks if the guessed letter is in the word
            if (!repeated) {
                int times = 0; //number of times a letter is in the word
                for (int index = 0; index < length; index++) {
                    if (wordToGuessChars[index] == currentGuessChar) {
                        censor[index] = currentGuessChar;  //replaces _ with guessed letter in caps
                        correct = true;
                        times++;
                    }
                }
                if (correct) {
                    System.out.println("The letter " + currentGuessChar + " is in the secret word! There are
" + times + " " + currentGuessChar + " 's in the word. Revealing the letter(s): ");
                } else {
                    System.out.println("Sorry, the letter is not in the word. Your secret word:  ");
                }
                System.out.println();
            }
            chances++;
        }
        System.out.println("You guessed the entire word " + wordToGuess.toUpperCase() + "
correctly! It took you " + chances + " attempts!");
    }
}
```

## Q2: Refactor your code – Code smell can give indications that there is some issue with the codes and can be solved by refactoring.

### Code Smells

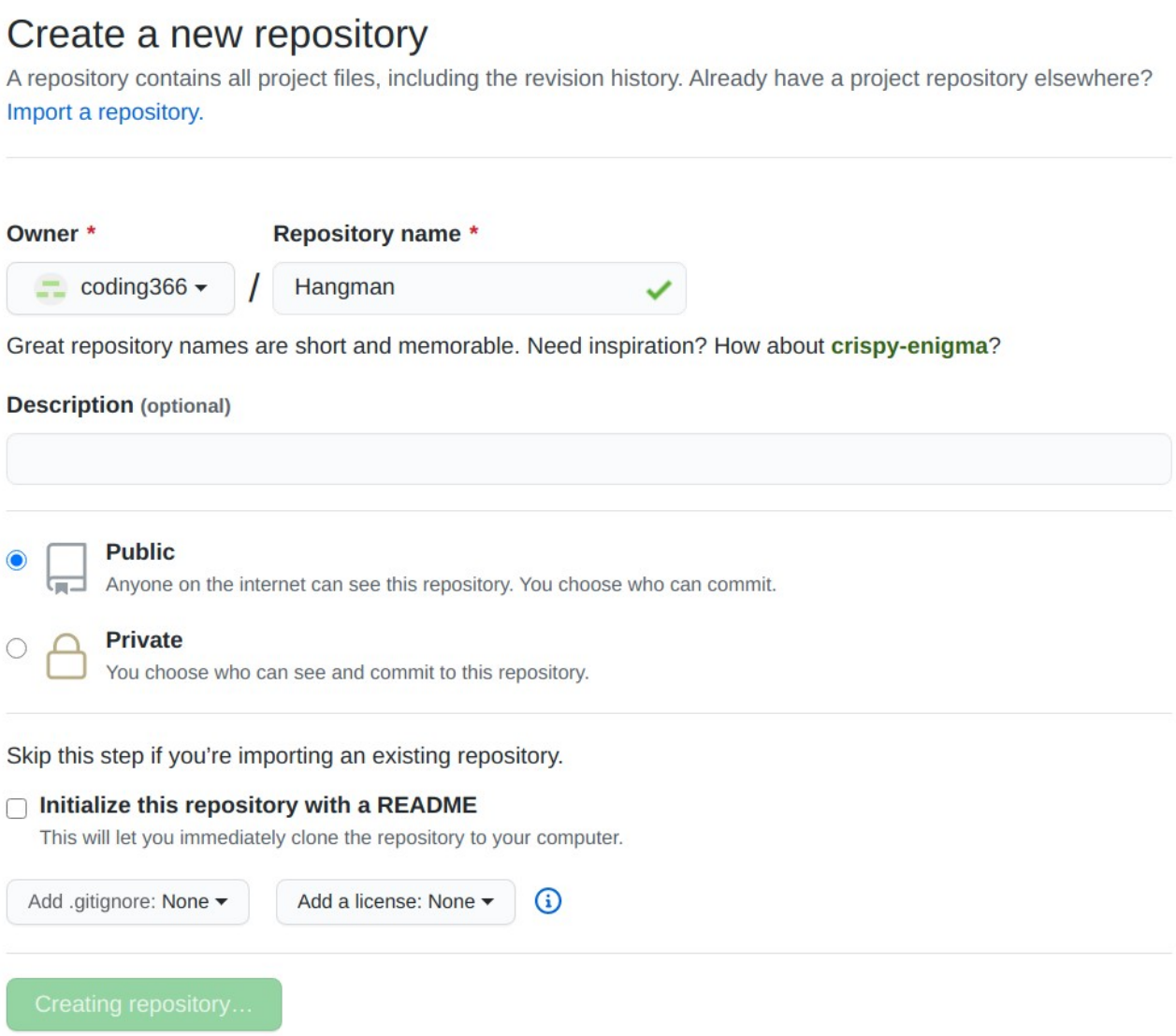
The names of the variables should always complete. The variables are self-explained so that these does not become mislead or confusing. I have make the separate method for generating the random numbers. Now the methods and variables are self-explainatory.

### Updated Code:

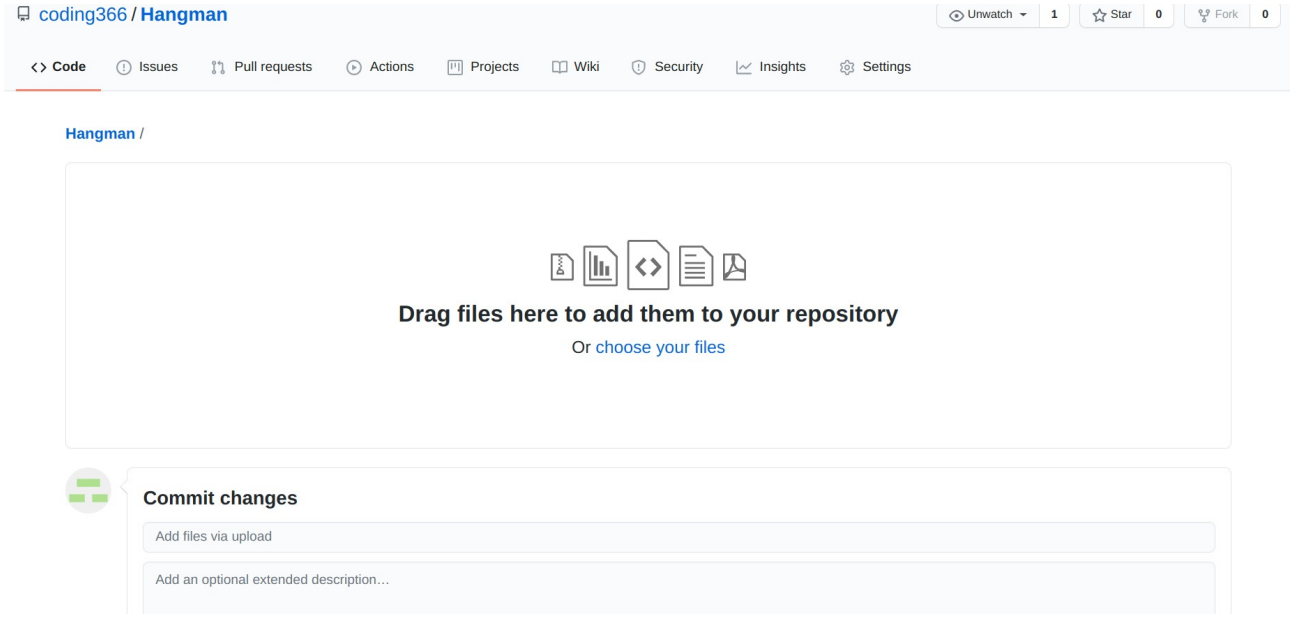
```
import java.util.HashSet;
import java.util.Random;
import java.util.Scanner;
import java.util.Set;
public class HangManClass {

    public static void main(String[] args) {
        guessWord(generateRandomWord());
    }

    private static String generateRandomWord(){
        int leftLimit = 97;
        int rightLimit = 122;
        int targetStringLength = 4;
        Random random = new Random();
        return  random.ints(leftLimit, rightLimit + 1)
                .limit(targetStringLength)
                .collect(StringBuilder::new, StringBuilder::appendCodePoint, StringBuilder::append)
                .toString();
    }

    private static void guessWord(String wordToGuess) {
        int attempts = 0;
        Set<String> previousGuesses = new HashSet<>();
        Scanner input = new Scanner(System.in);
        int wordLength = wordToGuess.length();
        char[] wordToGuessChars = wordToGuess.toCharArray();
        char[] censor = wordToGuess.toCharArray();
        System.out.println("Your secret word is : "+wordToGuess);
        for (int index = 0; index < wordLength; index++) {
            censor[index] = '_';
        }
        while (!String.valueOf(censor).equals(wordToGuess)) {
            boolean isCorrect = false;
            boolean isRepeated = false;
            for (int a = 0; a < wordLength; a++) {
                System.out.print(censor[a]);
            }
            System.out.println();
            System.out.print("Type your guess: ");
```

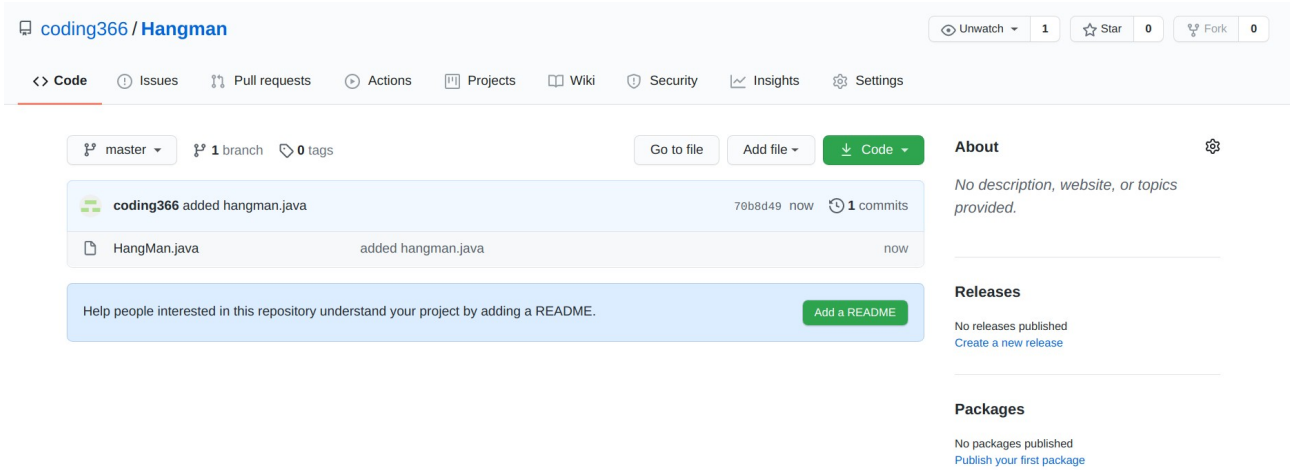
```
            String currentGuess = input.next().substring(0, 1);
            char currentGuessCharacter = currentGuess.charAt(0);
            if (previousGuesses.contains(currentGuess))
            {
                System.out.println("You already guessed this letter! Try again. Your previous guesses
were: ");
                System.out.println(previousGuesses.stream().reduce("", String::concat));
                isRepeated = true;
            }
            previousGuesses.add(currentGuess);
            if (!isRepeated) {
                int times = 0;
                for (int index = 0; index < wordLength; index++) {
                    if (wordToGuessChars[index] == currentGuessCharacter) {
                        censor[index] = currentGuessCharacter;
                        isCorrect = true;
                        times++;
                    }
                }
                if (isCorrect) {
                    System.out.println("The letter " + currentGuessCharacter + " is in the secret word!
There are " + times + " " + currentGuessCharacter + " 's in the word.");
                } else {
                    System.out.println("Sorry, the letter is not in the word. Your secret word: ");
                }
                System.out.println();
            }
            attempts++;
        }
        System.out.println("You guessed the entire word " + wordToGuess.toUpperCase() + "
correctly! It took you " + attempts + " attempts!");
    }

    @Override
    public String toString() {
        return "HangManClass{}";
    }
}
```

**Q3: Create a Git directory for your assignment (including word or pdf documents and programming code)**




Link to the Github Repository : https://github.com/coding366/Hangman


coding366 / Hangman
Unwatch 1
Star 0
Fork 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Hangman /
Drag files here to add them to your repository
Or choose your files
Commit changes
Add files via upload
Add an optional extended description…


coding366 / Hangman
Unwatch 1
Star 0
Fork 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
master
1 branch
0 tags
Go to file
Add file
Code
coding366 added hangman.java
70b8d49 now
1 commits
HangMan.java
added hangman.java
now
Help people interested in this repository understand your project by adding a README.
Add a README
About
No description, website, or topics provided.
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package

master **Hangman** / **HangMan.java** / Jump to Go to file ···

**coding366** added hangman.java Latest commit 70b8d49 28 seconds ago **History**

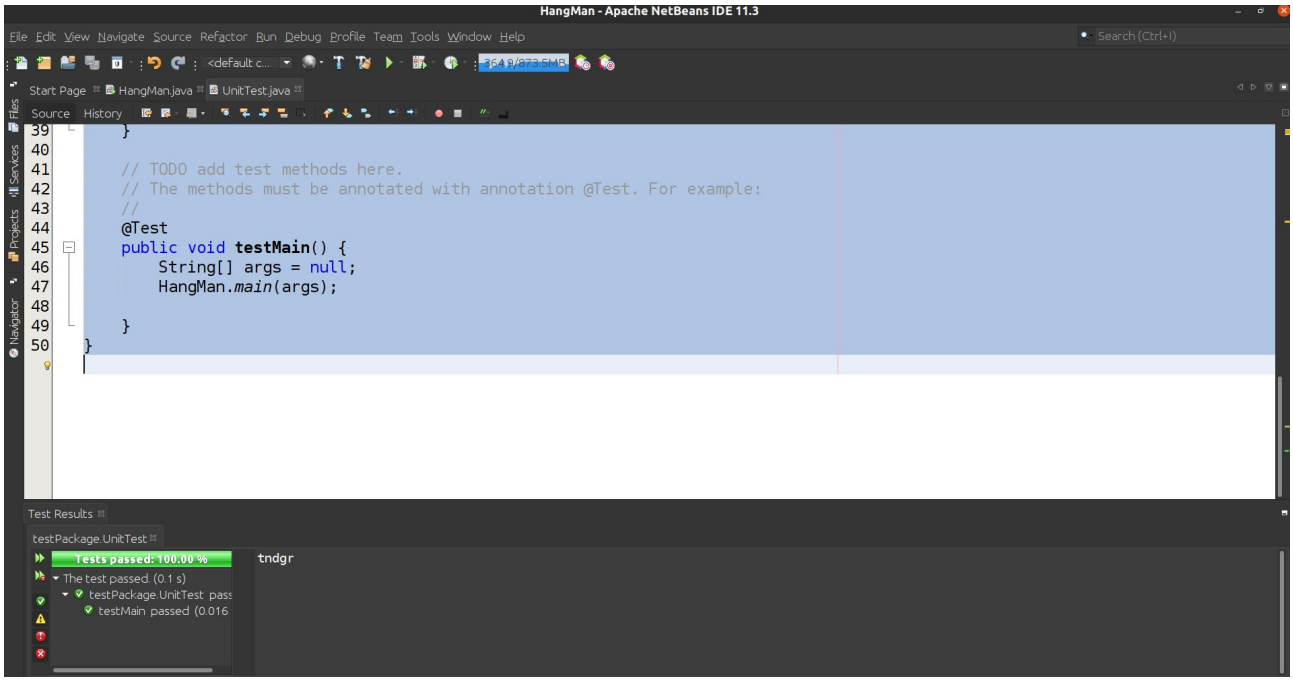
**1** contributor

72 lines (67 sloc) | 3.34 KB Raw Blame

```java
package hangman;

import java.util.HashSet;
import java.util.Random;
import java.util.Set;
import java.util.Scanner;

class Hangman {

    public static void main(String[] args) {
        int chances = 0;
        Set<String> prevGuesses = new HashSet<>();
        Scanner input = new Scanner(System.in);
        int leftLimit = 97; // letter 'a'
        int rightLimit = 122; // letter 'z'
        int targetStringLength = 5;//length of the word
        Random random = new Random();
 //generating random word using pre-built java8 method.
    String wordToGuess = random.ints(leftLimit, rightLimit + 1)
      .limit(targetStringLength)
      .collect(StringBuilder::new, StringBuilder::appendCodePoint, StringBuilder::append)
```

## Q No.4

**a) how TDD has been implemented to create your program.**
We have used the TDD approach for the development of this application. We could easily test using the method of Random Generator to check that a random no of specified is being generated or not. The screenshot of the tests are being attached.




## Code :

```
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;

public class HangManClassTest {

    public HangManClassTest() {
    }

    @BeforeClass
    public static void setUpClass() {
    }

    @AfterClass
    public static void tearDownClass() {
    }

    @Before
```

```
    public void setUp() {
    }

    @After
    public void tearDown() {
    }

    /**
     * Test of main method, of class HangManClass.
     */
    @Test
    public void testMain() {
        System.out.println("main");
        String[] args = null;
        HangManClass.main(args);
        // TODO review the generated test code and remove the default call to fail.
        fail("The test case is a prototype.");
    }

    /**
     * Test of toString method, of class HangManClass.
     */
    @Test
    public void testToString() {
        System.out.println("toString");
        HangManClass instance = new HangManClass();
        String expResult = "";
        String result = instance.toString();
        assertEquals(expResult, result);
        // TODO review the generated test code and remove the default call to fail.
        fail("The test case is a prototype.");
    }


    @Test
    public void testRandomGenerator() {
        System.out.println("Random Generator");
        HangManClass instance = new HangManClass();
        instance.guessWord(instance.generateRandomWord());
        // TODO review the generated test code and remove the default call to fail.
        fail("The test case is a prototype.");
    }
}
```

**b) How refactoring has been done, the code smell, issues and solutions.**

The refactoring of the code is done and also the latest code committed to github repository. You could check the commit history. The issue were that the variables were not self-explanatory and all the code is the main method. If any problem occurs then I have to debug complete code to find out the issue. But now different methods are formed according to their functions. Now the methods and variables are well-defined.