GEBZE TECHNICAL UNIVERSITY

CSE 222
DATA STRUCTURES AND ALGORITHMS

HOMEWORK 4 REPORT

OGUZ MUTLU
1801042624

1. DETAILED SYSTEM REQUIREMENTS

This homework design a securtiy system for the museum in Topkapi palace.The museum security system must provide an encrypted password for each museum officer.

The officer types three following inputs to the security system.

Username (String) only letter included minimum length must be at least one.

Password1(String) contains only letter and brackets each string password must contain at least 2 brackets. The minimum length must be 8 including brackets

Password2(int) number must between 10 and 10000

Checking username validity must be done with recursive functions, and some check operations like username spirit must use stack data structure. We have to check also given string is convertible to the palindrome structure when we doing this check opration we have to use stack and recursion together.

In password2 we have to check exact division by given denominations we have to do this with using recursion.

3. PROBLEM SOLUTION APPROACH

I create 4 different class one is for testing other classes. My classes;

- Username
- PasswordOne
- PasswordTwo
- EncryptionTest

First I check username character validity with string functions and recursive with the help of **checkIfValidUsername**(String username) function this function throws NoSuchElement exception when string is null. This function check if username is valid include character or not This function check username if username current character is valid send substring of username (1 to end of username)

Then I check the **userNameSpirit** of the password this function checks the given password includes at least one character of username.
This function checks the password includes the user name letter i send all password characters into the passwordstack data structure and in every sent i check if password and username includes same letter. Also check password contains any digit number if it includes digit number return false otherwise return true

Checking is balanced; I design that function by using stack
This function check balanced or not of the password string This function pushes all of the brackets in the stack any letter or digit value is not pushed into the stack Then close brackets observed this function check whether last stack member which is pop() is same type of the bracketssame type but opening bracket if it is true stack delete openin brackets and so on. At the end if stack empty returns true otherwise false

In **palindrome** part of the password 1 I write a function recursively with helper function this function takes string, mutable string, size, and flag variables

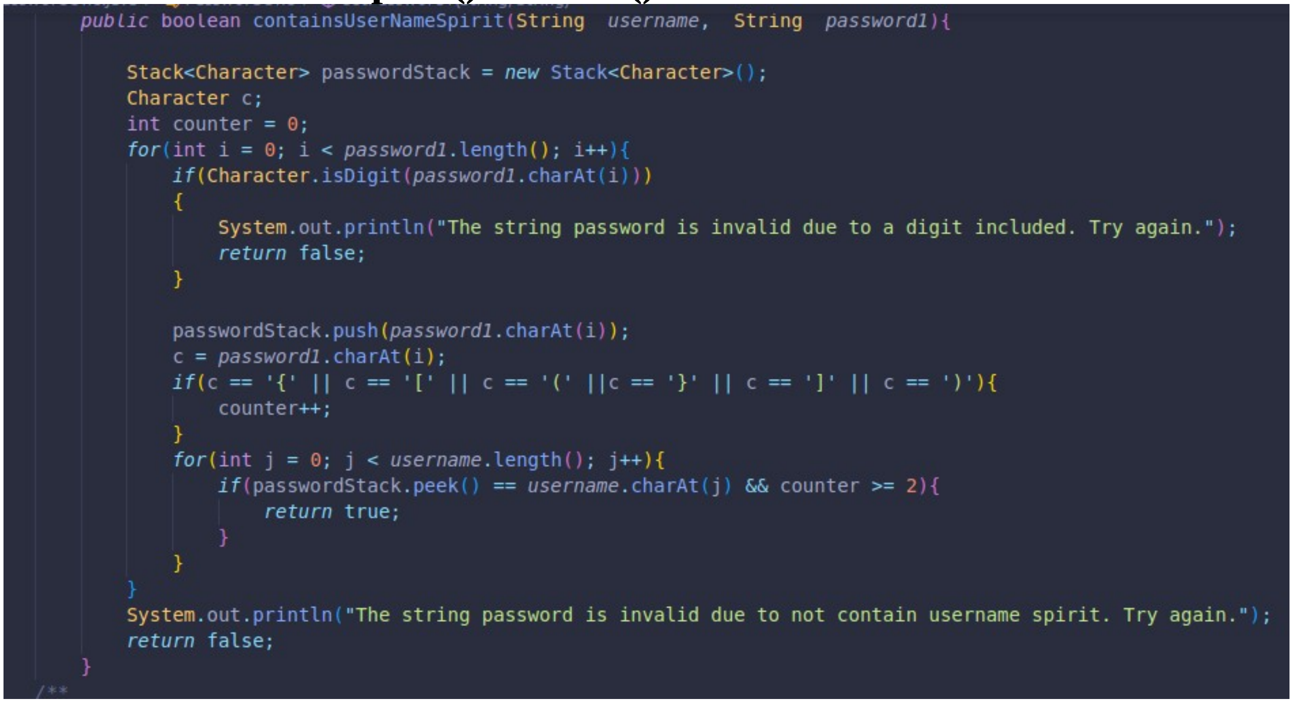
- This function is used to check if the password convertible to palindrome structure
- First of all index set to the at the end of password string and divide from 0 to until that character index substring created function checks the character through this substring.
- If it is in the first index function just invokes itself and decrease its size
- Then second from the end of password string will check the character through this substring but this substring is not not include first and current index (so if index = 5 and length = 6 substring includes (1, 4) characters)
- if string found in index>0 that we make simple change in that characters with using index and length -size math operations
- if there are one character which has length is 1this function set this character into to the middle of the mutable string and returns the same index
- if there are more than one character which length is 1 flag control this character and returns true if more than one

When checking **exact division** I design a function recursively my function Check if the password exact divisibile by denominators

- if denominors last higher than the password then i check remain part is 0 or not
- then if it is not i decrease index of denominators and do same check with that index of denom array
- if all true then this function returns true

Time Complexity Analysis

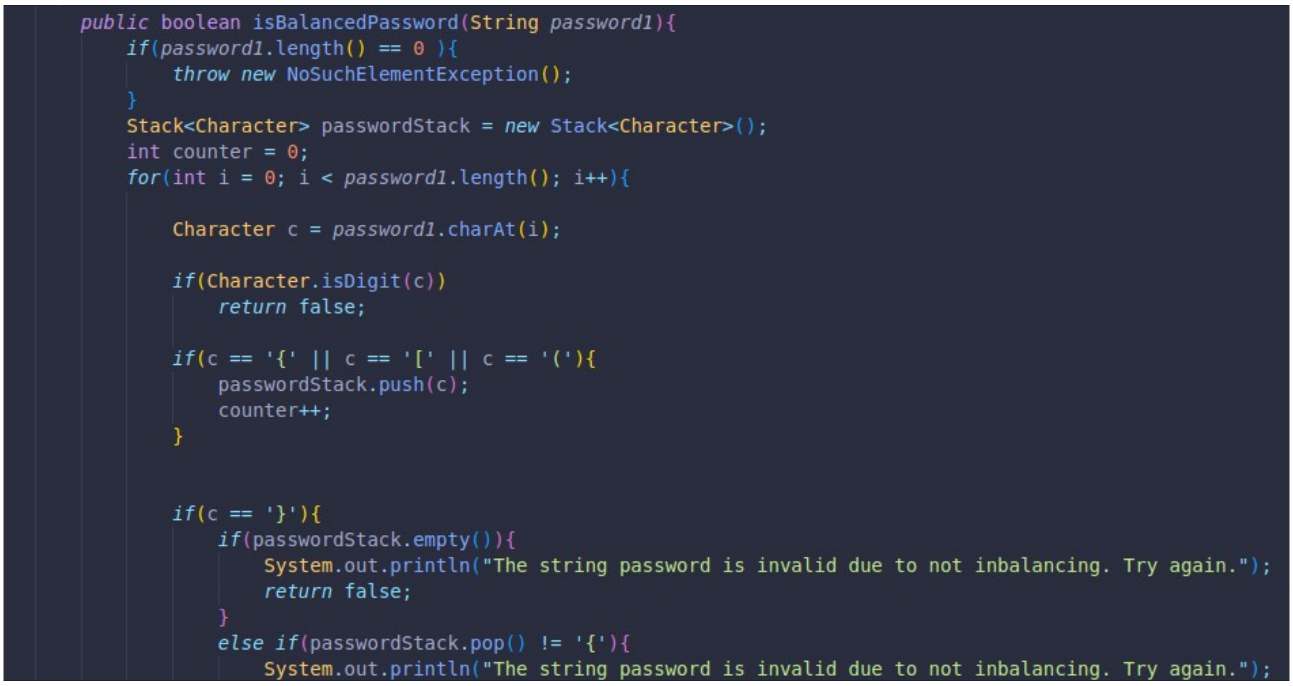
containsUserNameSpirit() method()



```java
    public boolean containsUserNameSpirit(String  username,  String  password1){

        Stack<Character> passwordStack = new Stack<Character>();
        Character c;
        int counter = 0;
        for(int i = 0; i < password1.length(); i++){
            if(Character.isDigit(password1.charAt(i)))
            {
                System.out.println("The string password is invalid due to a digit included. Try again.");
                return false;
            }

            passwordStack.push(password1.charAt(i));
            c = password1.charAt(i);
            if(c == '{' || c == '[' || c == '(' ||c == '}' || c == ']' || c == ')'){
                counter++;
            }
            for(int j = 0; j < username.length(); j++){
                if(passwordStack.peek() == username.charAt(j) && counter >= 2){
                    return true;
                }
            }
        }
        System.out.println("The string password is invalid due to not contain username spirit. Try again.");
        return false;
    }
```

Initializations are = O(1)
for loop = O(n)
push method = O(1)
charAt method = O(1)
inner for loop = O(m)
Total Complexity = O(n^2)
NOTE : Checks both username and password inside so we have to design inner loop that means n^2 complexity occurs.
---

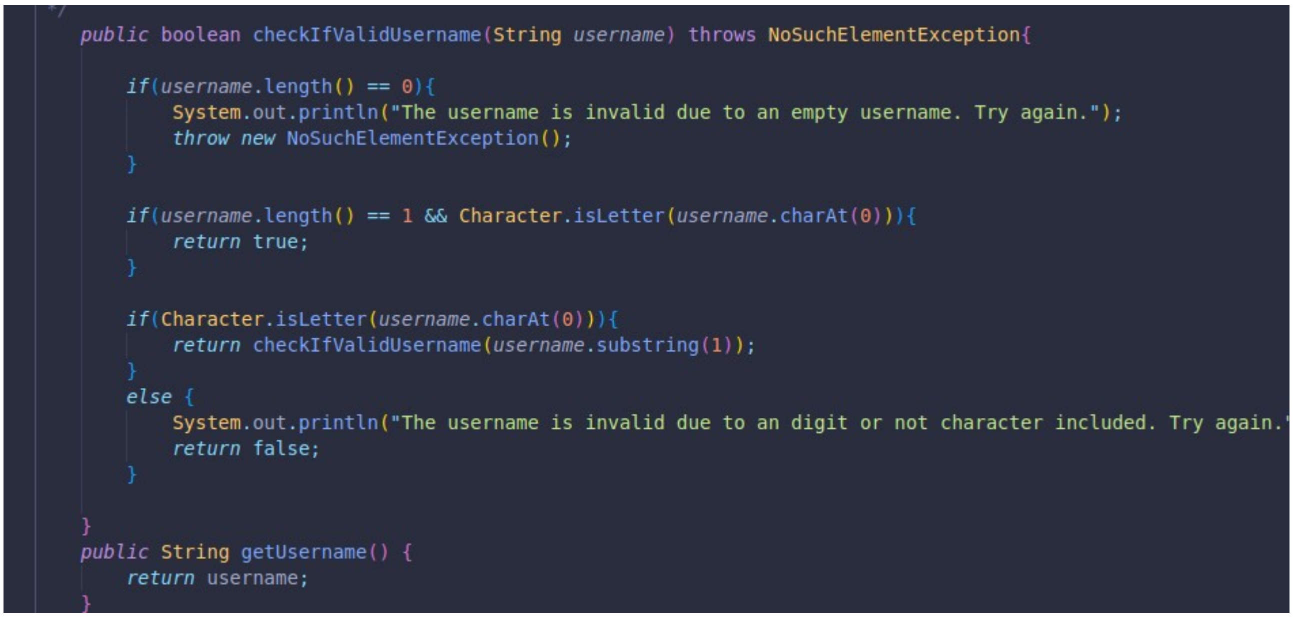
isBalancedPassword() Method



```java
public boolean isBalancedPassword(String password1){
    if(password1.length() == 0 ){
        throw new NoSuchElementException();
    }
    Stack<Character> passwordStack = new Stack<Character>();
    int counter = 0;
    for(int i = 0; i < password1.length(); i++){

        Character c = password1.charAt(i);

        if(Character.isDigit(c))
            return false;

        if(c == '{' || c == '[' || c == '('){
            passwordStack.push(c);
            counter++;
        }


        if(c == '}'){
            if(passwordStack.empty()){
                System.out.println("The string password is invalid due to not inbalancing. Try again.");
                return false;
            }
            else if(passwordStack.pop() != '{'){
                System.out.println("The string password is invalid due to not inbalancing. Try again.");
```

Stack initialization and counter initialization = O(1)
password.length() for loop = O(n)
push method = O(1)
isDigit() method = O(1)
if comparisons = O(1)
stack empty() method = O(1)
Total Complexity = O(n)
For loop iterates over length of password and other stack functions does not any length iteration so whole function complexity is O(n)

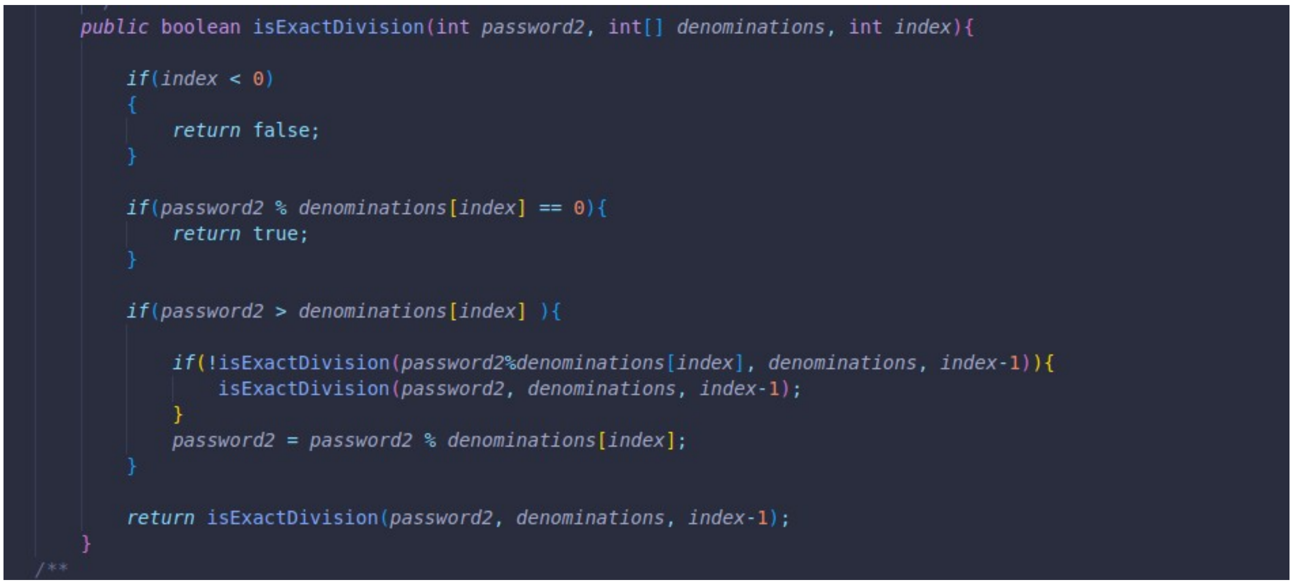
checkValidUsername() method



```
*/
public boolean checkIfValidUsername(String username) throws NoSuchElementException{

    if(username.length() == 0){
        System.out.println("The username is invalid due to an empty username. Try again.");
        throw new NoSuchElementException();
    }

    if(username.length() == 1 && Character.isLetter(username.charAt(0))){
        return true;
    }

    if(Character.isLetter(username.charAt(0))){
        return checkIfValidUsername(username.substring(1));
    }
    else {
        System.out.println("The username is invalid due to an digit or not character included. Try again."
        return false;
    }

}
public String getUsername() {
    return username;
}
```

If the length of the string is 1 best case = **O(1)**
Thus we keep index of the character such as substring() we invoke function over string by increase index by 1
so that means our recursive function process **O(n)** times
Total complexity is O(n)
--

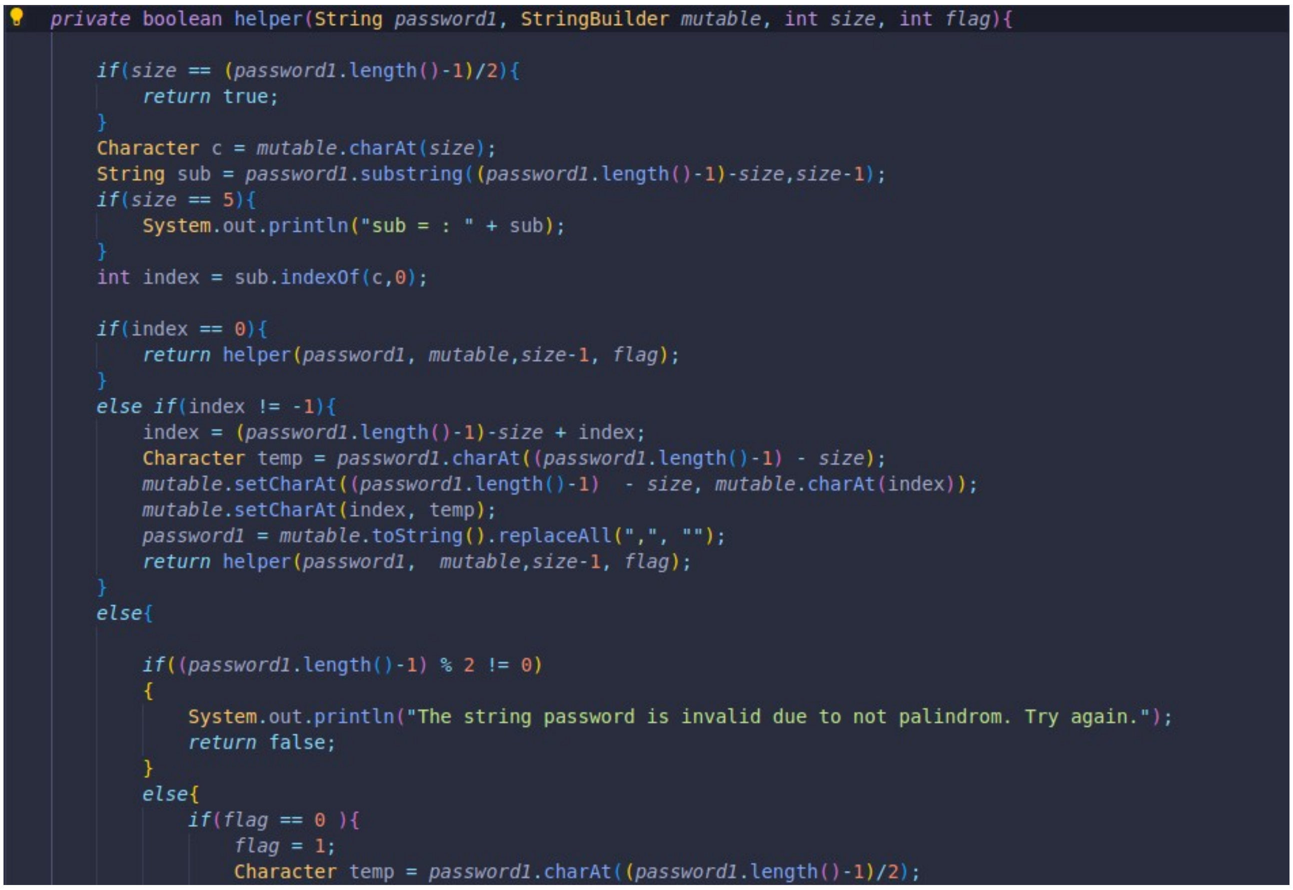
isExactDivision() method



```java
    public boolean isExactDivision(int password2, int[] denominations, int index){

        if(index < 0)
        {
            return false;
        }

        if(password2 % denominations[index] == 0){
            return true;
        }

        if(password2 > denominations[index] ){

            if(!isExactDivision(password2%denominations[index], denominations, index-1)){
                isExactDivision(password2, denominations, index-1);
            }
            password2 = password2 % denominations[index];
        }

        return isExactDivision(password2, denominations, index-1);
    }
/**
```

If the denominators are in sorted then this function iterates the index starting from the last index to the first index worst case iterates length of denominations lengths so that means O(n) other modulation if statements and assignment operation all done in O(1) time
our function iterates over
1. isExactDivision(psw2, denomination,3)
2. isExactDivision(psw2, denomination,2)
3. isExactDivision(psw2, denomination,1)
index number decreases
So total time complexity is O(n)

isPalindromePossible() and helper() methods



```java
private boolean helper(String password1, StringBuilder mutable, int size, int flag){

    if(size == (password1.length()-1)/2){
        return true;
    }
    Character c = mutable.charAt(size);
    String sub = password1.substring((password1.length()-1)-size,size-1);
    if(size == 5){
        System.out.println("sub = : " + sub);
    }
    int index = sub.indexOf(c,0);

    if(index == 0){
        return helper(password1, mutable,size-1, flag);
    }
    else if(index != -1){
        index = (password1.length()-1)-size + index;
        Character temp = password1.charAt((password1.length()-1) - size);
        mutable.setCharAt((password1.length()-1)  - size, mutable.charAt(index));
        mutable.setCharAt(index, temp);
        password1 = mutable.toString().replaceAll(",", "");
        return helper(password1,  mutable,size-1, flag);
    }
    else{

        if((password1.length()-1) % 2 != 0)
        {
            System.out.println("The string password is invalid due to not palindrom. Try again.");
            return false;
        }
        else{
            if(flag == 0 ){
                flag = 1;
                Character temp = password1.charAt((password1.length()-1)/2);
```

This function first extracts all brackets from the password that means O(n) in we iterate all over the password. Then we call helper function. CharAt function in O(1) time and string substring function iterates all over the n-1 length that means O(n) at most my recursive function iterates all over the string from beginning to end of the string first last string will be consider and according the the last string character first string character will be set that means if we consider last character we set first character, if we consider (last – 1)th character that means we set 1st character, if we consider (last – 2)th character 2nd character and so on… that means we iterate recursive function O(n) times and inside this function substring control operations done in O(n) time so this functions total time complexity is **O(n^2)**

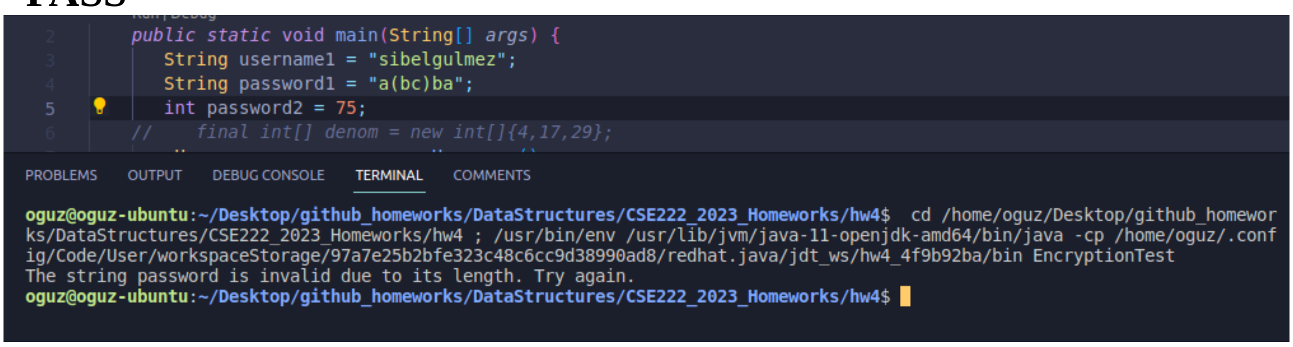
4.TEST CASES AND RESULTS

Case -1 Steps

1. username=sibelgulmez
2. password1 = a(bc)ba
3. password2 = 75

Expectation:
- string password length at least 8

Result
- PASS



```java
public static void main(String[] args) {
    String username1 = "sibelgulmez";
    String password1 = "a(bc)ba";
    int password2 = 75;
//    final int[] denom = new int[]{4,17,29};
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

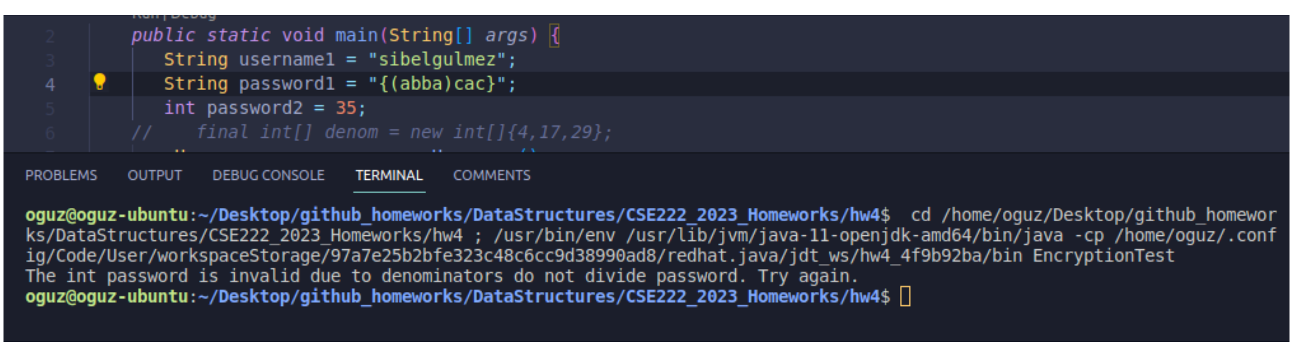
```
oguz@oguz-ubuntu:~/Desktop/github_homeworks/DataStructures/CSE222_2023_Homeworks/hw4$  cd /home/oguz/Desktop/github_homewor
ks/DataStructures/CSE222_2023_Homeworks/hw4 ; /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java -cp /home/oguz/.conf
ig/Code/User/workspaceStorage/97a7e25b2bfe323c48c6cc9d38990ad8/redhat.java/jdt_ws/hw4_4f9b92ba/bin EncryptionTest
The string password is invalid due to its length. Try again.
oguz@oguz-ubuntu:~/Desktop/github_homeworks/DataStructures/CSE222_2023_Homeworks/hw4$
```

Case -2 Steps

1. username=sibelgulmez
2. password1 = {(abba)cac}
3. password2 = 35

Expectation:
- Denominators Do not divide password2

Result
- PASS



```java
public static void main(String[] args) {
    String username1 = "sibelgulmez";
    String password1 = "{(abba)cac}";
    int password2 = 35;
//    final int[] denom = new int[]{4,17,29};
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

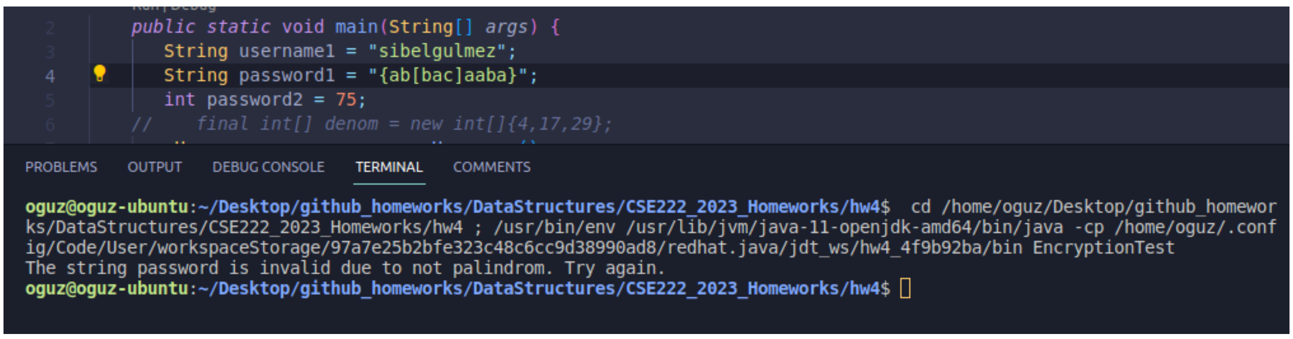
```
oguz@oguz-ubuntu:~/Desktop/github_homeworks/DataStructures/CSE222_2023_Homeworks/hw4$  cd /home/oguz/Desktop/github_homewor
ks/DataStructures/CSE222_2023_Homeworks/hw4 ; /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java -cp /home/oguz/.conf
ig/Code/User/workspaceStorage/97a7e25b2bfe323c48c6cc9d38990ad8/redhat.java/jdt_ws/hw4_4f9b92ba/bin EncryptionTest
The int password is invalid due to denominators do not divide password. Try again.
oguz@oguz-ubuntu:~/Desktop/github_homeworks/DataStructures/CSE222_2023_Homeworks/hw4$
```

Case -3 Steps

1. username=sibelgulmez
2. password1 = {ab[bac]aaba}
3. password2 = 75

Expectation:
- Unsuccesfull due to palindrom

Result
- PASS



```java
public static void main(String[] args) {
    String username1 = "sibelgulmez";
    String password1 = "{ab[bac]aaba}";
    int password2 = 75;
//    final int[] denom = new int[]{4,17,29};
```

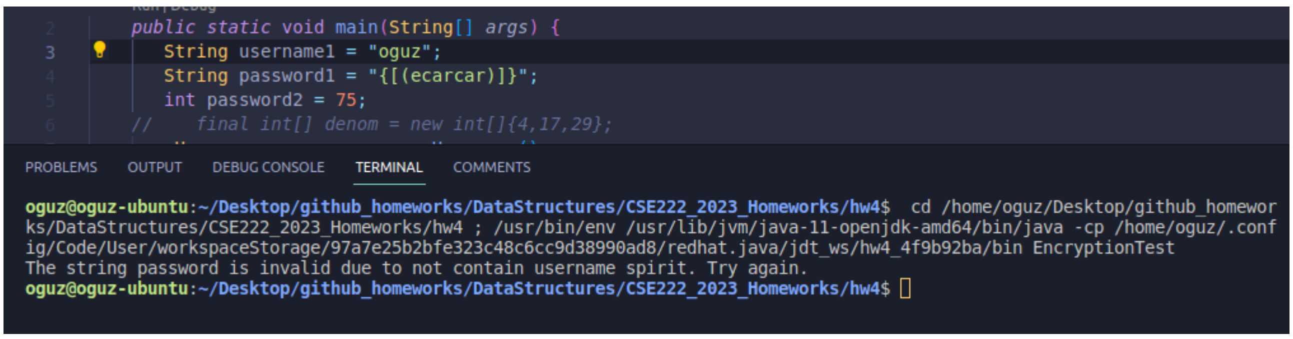
```
oguz@oguz-ubuntu:~/Desktop/github_homeworks/DataStructures/CSE222_2023_Homeworks/hw4$  cd /home/oguz/Desktop/github_homewor
ks/DataStructures/CSE222_2023_Homeworks/hw4 ; /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java -cp /home/oguz/.conf
ig/Code/User/workspaceStorage/97a7e25b2bfe323c48c6cc9d38990ad8/redhat.java/jdt_ws/hw4_4f9b92ba/bin EncryptionTest
The string password is invalid due to not palindrom. Try again.
oguz@oguz-ubuntu:~/Desktop/github_homeworks/DataStructures/CSE222_2023_Homeworks/hw4$ 
```

Case -4 Steps

1. username= oguz
2. password1 = {[(ecarcar)]}
3. password2 = 75

Expectation:
- Unsuccesfull due to username spirit

Result
- PASS



```java
public static void main(String[] args) {
    String username1 = "oguz";
    String password1 = "{[(ecarcar)]}";
    int password2 = 75;
//    final int[] denom = new int[]{4,17,29};
```

```
oguz@oguz-ubuntu:~/Desktop/github_homeworks/DataStructures/CSE222_2023_Homeworks/hw4$  cd /home/oguz/Desktop/github_homewor
ks/DataStructures/CSE222_2023_Homeworks/hw4 ; /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java -cp /home/oguz/.conf
ig/Code/User/workspaceStorage/97a7e25b2bfe323c48c6cc9d38990ad8/redhat.java/jdt_ws/hw4_4f9b92ba/bin EncryptionTest
The string password is invalid due to not contain username spirit. Try again.
oguz@oguz-ubuntu:~/Desktop/github_homeworks/DataStructures/CSE222_2023_Homeworks/hw4$ 
```

Case -5 Steps

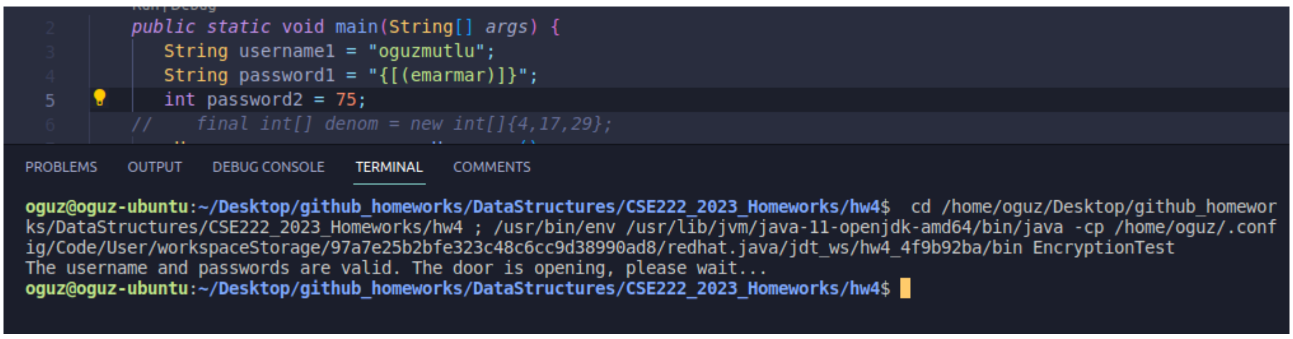
1. username= oguzmutlu
2. password1 = {[(emarmar)]}
3. password2 = 75

Expectation:

- Succesfully opens the door

Result

- PASS



```
public static void main(String[] args) {
    String username1 = "oguzmutlu";
    String password1 = "{[(emarmar)]}";
    int password2 = 75;
//    final int[] denom = new int[]{4,17,29};
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

```
oguz@oguz-ubuntu:~/Desktop/github_homeworks/DataStructures/CSE222_2023_Homeworks/hw4$  cd /home/oguz/Desktop/github_homewor
ks/DataStructures/CSE222_2023_Homeworks/hw4 ; /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java -cp /home/oguz/.conf
ig/Code/User/workspaceStorage/97a7e25b2bfe323c48c6cc9d38990ad8/redhat.java/jdt_ws/hw4_4f9b92ba/bin EncryptionTest
The username and passwords are valid. The door is opening, please wait...
oguz@oguz-ubuntu:~/Desktop/github_homeworks/DataStructures/CSE222_2023_Homeworks/hw4$
```

Case -6 Steps

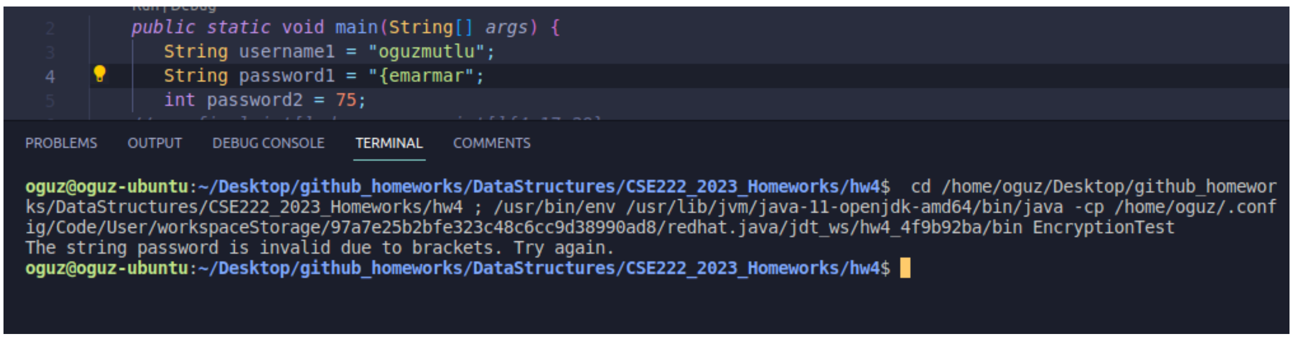
1. username= oguzmutlu
2. password1 = {emarmar
3. password2 = 75

Expectation:

- The brackets have to be at least 2

Result

<u>- PASS</u>



```
public static void main(String[] args) {
    String username1 = "oguzmutlu";
    String password1 = "{emarmar";
    int password2 = 75;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

```
oguz@oguz-ubuntu:~/Desktop/github_homeworks/DataStructures/CSE222_2023_Homeworks/hw4$  cd /home/oguz/Desktop/github_homewor
ks/DataStructures/CSE222_2023_Homeworks/hw4 ; /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java -cp /home/oguz/.conf
ig/Code/User/workspaceStorage/97a7e25b2bfe323c48c6cc9d38990ad8/redhat.java/jdt_ws/hw4_4f9b92ba/bin EncryptionTest
The string password is invalid due to brackets. Try again.
oguz@oguz-ubuntu:~/Desktop/github_homeworks/DataStructures/CSE222_2023_Homeworks/hw4$
```

Case -6 Steps

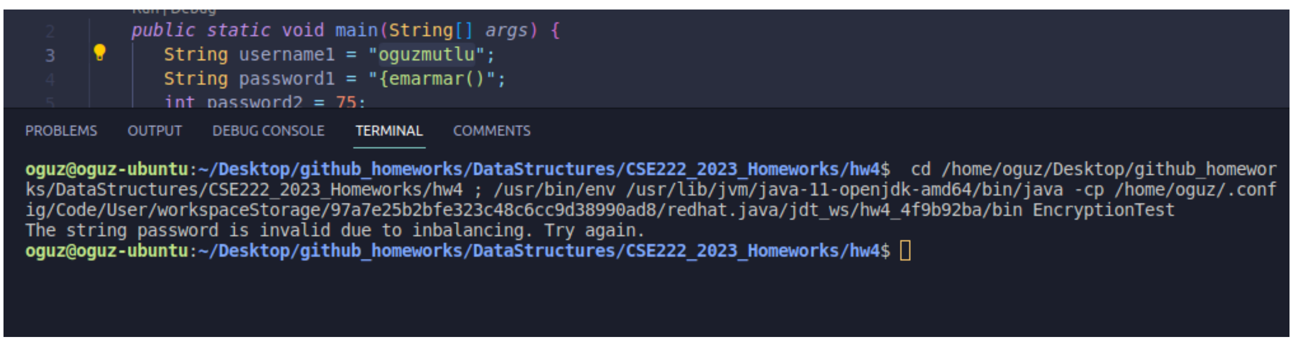
1. username= oguzmutlu
2. password1 = {emarmar()
3. password2 = 75

Expectation:

- The inbalancing problem occurs

Result

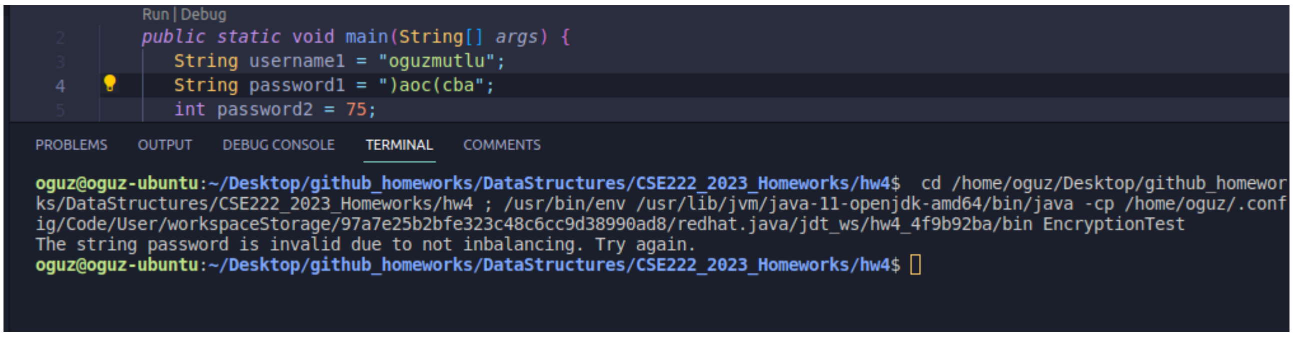
<u>- PASS</u>



```
public static void main(String[] args) {
    String username1 = "oguzmutlu";
    String password1 = "{emarmar()";
    int password2 = 75;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

```
oguz@oguz-ubuntu:~/Desktop/github_homeworks/DataStructures/CSE222_2023_Homeworks/hw4$  cd /home/oguz/Desktop/github_homewor
ks/DataStructures/CSE222_2023_Homeworks/hw4 ; /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java -cp /home/oguz/.conf
ig/Code/User/workspaceStorage/97a7e25b2bfe323c48c6cc9d38990ad8/redhat.java/jdt_ws/hw4_4f9b92ba/bin EncryptionTest
The string password is invalid due to inbalancing. Try again.
oguz@oguz-ubuntu:~/Desktop/github_homeworks/DataStructures/CSE222_2023_Homeworks/hw4$ 
```



Run | Debug

```
public static void main(String[] args) {
    String username1 = "oguzmutlu";
    String password1 = ")aoc(cba";
    int password2 = 75;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

```
oguz@oguz-ubuntu:~/Desktop/github_homeworks/DataStructures/CSE222_2023_Homeworks/hw4$  cd /home/oguz/Desktop/github_homewor
ks/DataStructures/CSE222_2023_Homeworks/hw4 ; /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java -cp /home/oguz/.conf
ig/Code/User/workspaceStorage/97a7e25b2bfe323c48c6cc9d38990ad8/redhat.java/jdt_ws/hw4_4f9b92ba/bin EncryptionTest
The string password is invalid due to not inbalancing. Try again.
oguz@oguz-ubuntu:~/Desktop/github_homeworks/DataStructures/CSE222_2023_Homeworks/hw4$ 
```