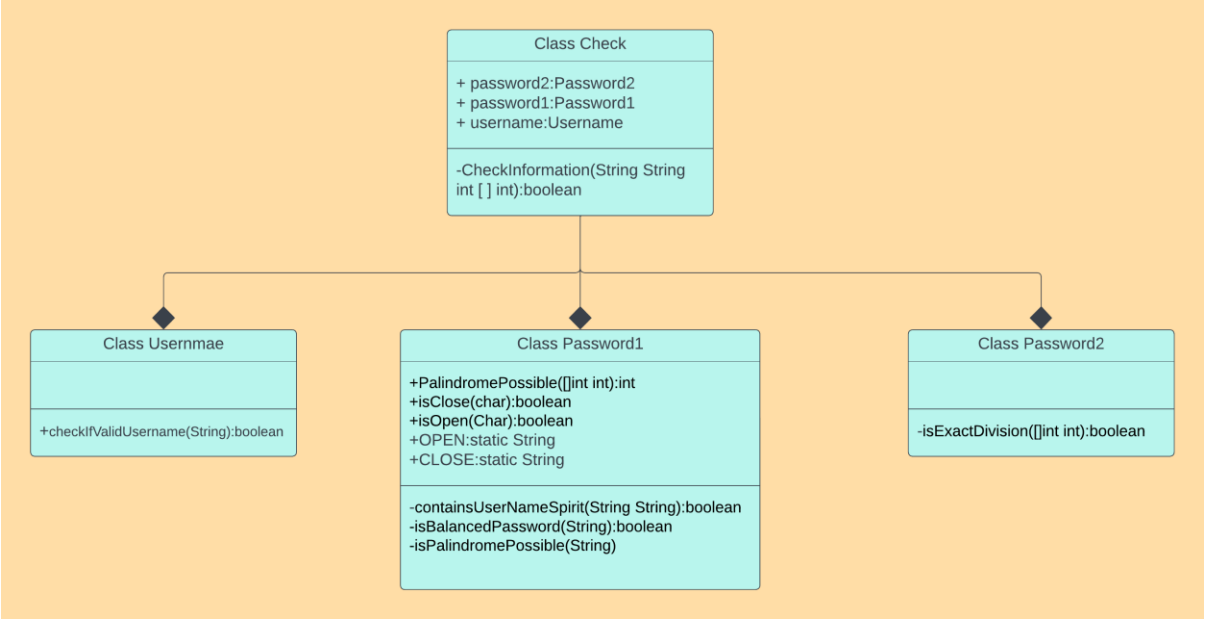
GIT Department of Computer Engineering
CSE 222/505 - Spring 2023
Homework # Report

Hamza Konaç
210104004202

1. **SYSTEM REQUIREMENTS**

   In this assignment we must to creaate a security system for topkapı palace. In this assignment there are 3 step for verify user.first step is verify user name. other steps about password. There are 2 password for system . one of them this is string and it need 3 requirement for verification. Another password has 1 requirement for verification.I have created 3 Class that named Username , Password1 Password2 for all of this 3 step . furthermore I have created extra one class for perform this stepds together.

2. **USE CASE AND CLASS DIAGRAMS**



3. PROBLEM SOLUTION APPROACH

Class Username:

checkIfValidUsername(String username):

In this step we need to check username.It must contain only.I my algoritm I have checked every Index recursively. If there is nonletter character in the username , false will be returned.

Time complexity:

T(n)=T(n-1)+O(n) ➔ T(n)=O(n^2)

T(0)= 1

->because of substring() methode we add O(n) to relation recurrence
T(0) to T(n) for requrrence means O(0) to O(n) ,
T(n)=O(n^2)

Class Password1:

containsUserNameSpirit(String username, String password1):

For this step we need to check password1 for contain any letter from username and use stack is required.In my algoritm I have created letter array for username and password.owing to this array I have taken letter from username and string.after that I have created 2 stack and push letter to stack respect alphabetical order.finaly I have check top element of the stack , ıf they equal I returned false.but ıf usrname top element is y and password stack's top element is 's' I poped from username stack.

Time complexity:

T(n)=O(n)

->for check letter and count them is linear.but stack them is constant .pop element or peek element is constant , too. because there is 25 letter in the alphabet . therefore stack's max size can be 25

isBalancedPassword(String password1) :

For this problem , I have created a stack. While checking string If there is open bracket , I pushed it into stack. If there is close bracket , I compare it with top element of the stack. If they don'r pair return false. End of the while loop If stack is not empty we return false.

Time complexity:

T(n)=O(n)

->compare stack is constant. Add element is constant.pop element is constant.check string is linear .

isPalindromePossible(String password1):

For this problem I Have created an array that measure frequency of each letter.this complexity is linear.after measure I have called helper function that calculate number of frequency that is odd.This helper funciton complexity is linear because there is 25 letter in alphabet therefore recursive run constant time for every input

Time complexity:

T(n)=O(n)

->measure freaquency is linear. Find item that has odd frequency is constant.

Class Password2:

isExactDivision(int password2, int [] denominations):

For this method I have created an algoritm that substract element of the denominations from password. After every substruction I have called function again with same denominations but new pasword that remain of the substract denominations's element from password2.

Time complexity:

T(n)=T(n)*T(n)* … T(n)

T(0)=1

T(n)=O(n^k)

->there is a loop in methode and its complexity is O(n).when we recursive funciton again we obtain new loop that complexity is O(n).Actually we have to try all possible to obtain password.for every element we can try n possibility and we have k element in the denominations array. So that complexity of funciton will be O(n^k)

4. TEST CASES

Username cases:

checkIfValidUsername

””

”h1mza”

”1hamza”

”hamz+”

”*amza”

Password1 cases:

containsUserNameSpirit

”hamza”->” {[(abacaba)]}”

”kerem”->” {[(abacaba)]}”

isBalancedPassword

”hamza”->” {[(abacaba)]}”

”hamza”->” {ab[bac]caba}

”hamza”->”)abc(cba”

”hamza”->” a]bcd(cb)a”

”hamza”->” [a]bcd(cb)a”

isPalindromePossible

”hamza”->” {[(ecarcar)]}”

”hamza”->” {ab[bac]aaba}””

”hamza”->” {(abba)cac}”

Password2 cases:

isExactDivision

”hamza”->” {(abba)cac}”->75->{4,17,29}

”hamza”->” {(abba)cac}”->74->{4,17,29}

”hamza”->” {(abba)cac}”->35->{4,17,29}

”hamza”->” {(abba)cac}”->54->{4,17,29}

5. RUNNING AND RESULTS

```
SECURITY SYSTEM

username="" password1="{[(abacaba)]}" password2="75" demoninations="{4 ,17,29}"

!!"user name must contain only letter"!! false

username="h1mza" password1="{[(abacaba)]}" password2="75" demoninations="{4 ,17,29}"

!!"user name must contain only letter"!! false

username="1hamza" password1="{[(abacaba)]}" password2="75" demoninations="{4 ,17,29}"

!!"user name must contain only letter"!! false

username="hamz+" password1="{[(abacaba)]}" password2="75" demoninations="{4 ,17,29}"

!!"user name must contain only letter"!! false

username="*amza" password1="{[(abacaba)]}" password2="75" demoninations="{4 ,17,29}"

!!"user name must contain only letter"!! false

username="hamza" password1="{[(abacaba)]}" password2="75" demoninations="{4 ,17,29}"

 true

username="kerem" password1="{[(abacaba)]}" password2="75" demoninations="{4 ,17,29}"

!!"password1 must contain least one letter from user name"!! false
```

```
username="hamza" password1="{[(abacaba)]}" password2="75" demoninations="{4 ,17,29}"

 true

username="hamza" password1="{ab[bac]caba}" password2="75" demoninations="{4 ,17,29}"

 true

username="hamza" password1=")abc(cba" password2="75" demoninations="{4 ,17,29}"

!!"password1 must contain balanced parathesis"!! false

username="hamza" password1="a]bcd(cb)a" password2="75" demoninations="{4 ,17,29}"

!!"password1 must contain balanced parathesis"!! false

username="hamza" password1="[a]bcd(cb)a" password2="75" demoninations="{4 ,17,29}"

 true

username="hamza" password1="{[(ecarcar)]}" password2="75" demoninations="{4 ,17,29}"

 true

username="hamza" password1="{ab[bac]aaba}" password2="75" demoninations="{4 ,17,29}"

!!"password1 must has possible palindrom string"!! false

username="hamza" password1="{(abba)cac}" password2="75" demoninations="{4 ,17,29}"

 true

username="hamza" password1="{(abba)cac}" password2="75" demoninations="{4 ,17,29}"

 true

username="hamza" password1="{(abba)cac}" password2="74" demoninations="{4 ,17,29}"

 true

username="hamza" password1="{(abba)cac}" password2="35" demoninations="{4 ,17,29}"

!!"password2 must to has exact divison variety of given demoninations"!! false

username="hamza" password1="{(abba)cac}" password2="54" demoninations="{4 ,17,29}"

 true
```