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

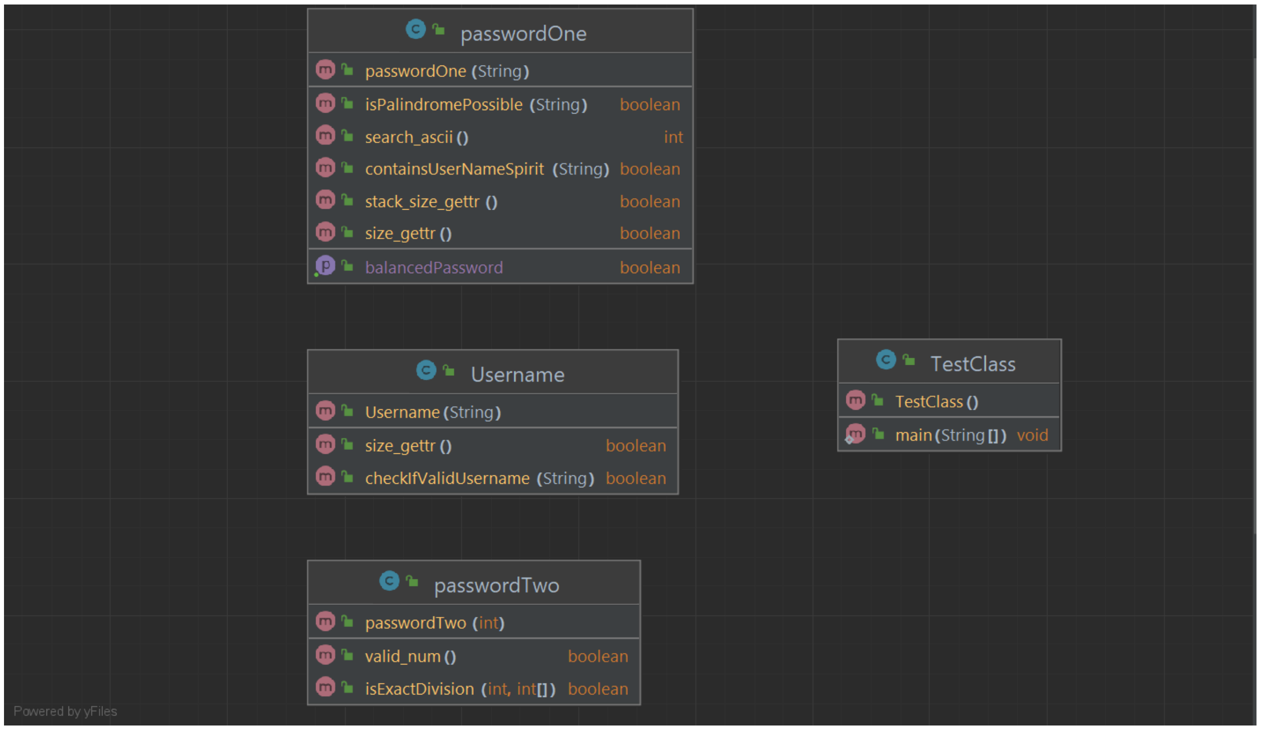
Yunus KARA

[redacted]

1. **SYSTEM REQUIREMENTS**

We wanted to design a security system. The museum security system must provide an encrypted password for each museum officer. There are two different password which names are password1 and password2.When officers types his/her informations our system has to decode both password and do some calculations. Officers also have an username and it has some rules. In this Project, things that was mentioned has to done by recursion methods or by using stack.

2. **USE CASE AND CLASS DIAGRAMS**



3. **OTHER DIAGRAMS**

Username Class:

- checkIfValidUsername methods travers through string recursively. It start from beginning to the end O(n).
- size_gettr methods which return length of username is just a gettr method so that's why time complexity is O(1)

passwordOne Class:

- containsUserNameSpirit takes username as a parameter of it. Then it starts searching for brackets. It uses stacks for every lettor for username. It starts from beginning to end with seperated two for nested O(n.m). N for password length and m for username length.
- isBalancedPassword method uses stack for pushing brackets. While pushing them, pop brackets. It is done linearly, O(n)
- isPolindromePossible method is a recursive method. It starts searching from beginning to end one by one. It doesn't care for any letters. O(n)
- search_ascii is a helper method for polindrome. It just goes through array for one time. There are 128 element in ascii table. O(128) so it is O(1).
- size_gettr and stack_size_gettr methods return just the value of the size. O(1).

passwordTwo.class:

- isExactDivision is a recursion function. Complexty of this algorithm so bad due to there is for. Every each calling functions runs for length of denominations. Length of denominations is L. Target is T. Best case is O(1) target and first element is the same. Worse case O(L^T). Complexity of this code is O(L^T).
- valid_num is a boolean method which checks the value is greater than ten and less then ten thousand. O(1)

4. PROBLEM SOLUTION APPROACH

For this project, username validation is done by username class. There is method for which name is "checkIfValidUsername". It is a recurisve method and it starts from beginning to end while doing this it calls itself and shrink the username from the start position. It should contains only letters otherwise it return false. Also, size is important if size is less then one, it return false. Meaning of the false is different. İf checkIfValidUsername return false, means that there is somethink that is not letter. If size_gettr is false means username is not valid because is less then one. For the passwordOne part, firstly, it checks is there at least one letter in password.
containsUserNameSpirit is for it. It push every letter that is in username then it pop from stack and compares them if they are same or not. If one of them is equal then it is true. Secondly, checking balanced password, there must exist brackets in the password. It uses stack for brackets if it is exist, it push them stack. If it finds ")}]" it pop from stack. However, stack is empty and trying to pop that means it is not balanced. Third part for isPalindromePossible, this method is a recursive method. It starts beginning of the string until reaching end. It creates substring array while calling itself. If it's find a brackets it didn't care about it. If there are any letter, it puts it an ascii_table array. Position is done by like this "a means 97" or "A means 65". When it reachs the end. İt calls search_ascii method and it search how many times they are occur. Returns number of odd numbers.

Being polindrome means. There are 2 rules. Any letter can only occur odd times or none. If length of password is less than 8 return false in size_gettr method because it helps password valid or not.Stack_size_gettr for brackets there must be at least two. Last part for passwordTwo, There are 2 method one of them is valid_num which checks for password greater or equalt to ten and less then ten thousand. Other method for isExactDivision. This is a recursive method. There is an for statement and it goes length of denominations. When it calls itself, it adds first element and it checks if it is equal to target or not. If it is then returns true. If not, then it adds until greater than target or being equalt. If greater then it substract what it is add. Then it adds next number and calls itself. It checks again.

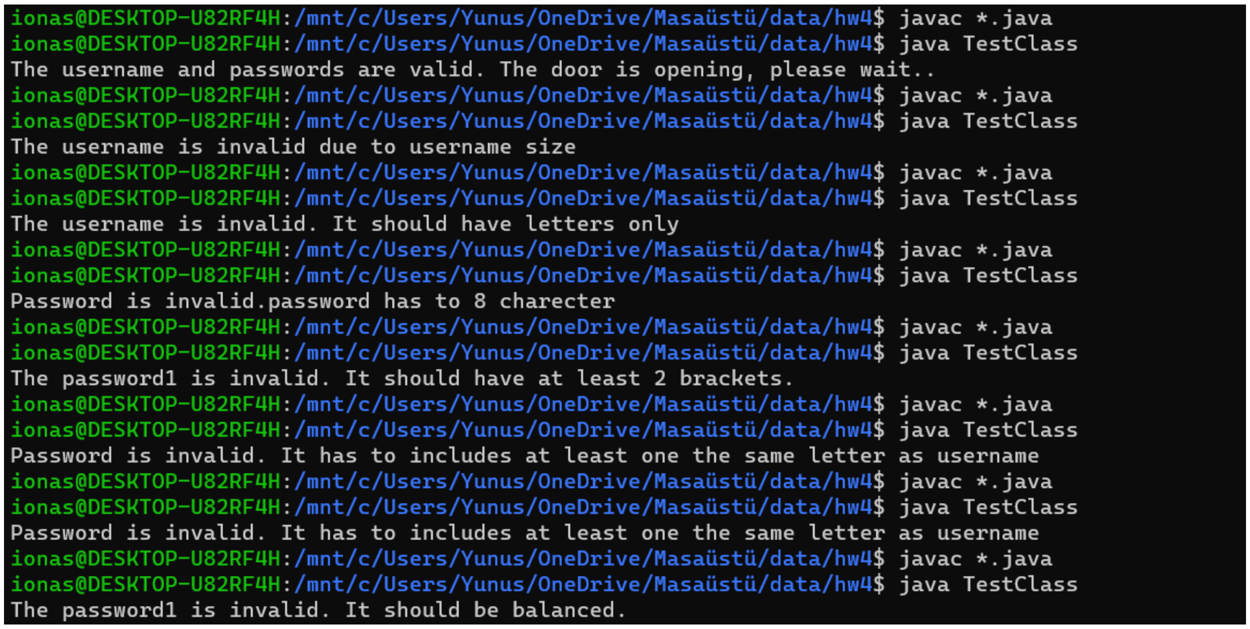
5. TEST CASES

Input and excepted output: *Test 1... Inputs:*
username: 'sibelgulmez' - password1: '[rac()ecar]' - password2: '74'
The username and passwords are valid. The door is opening, please wait..

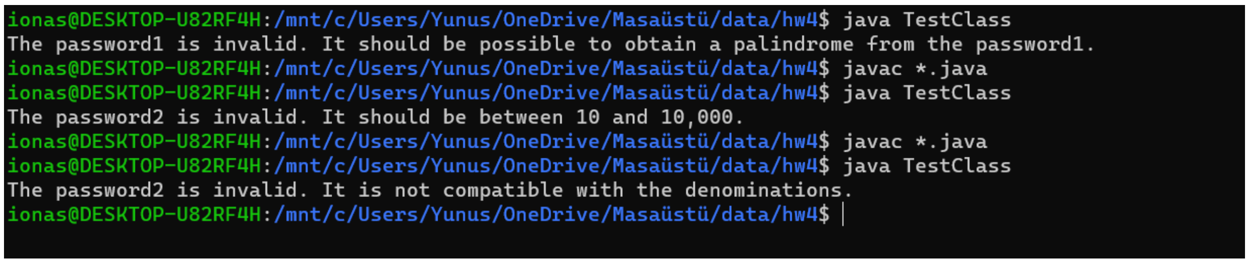
Test 2... Inputs:
username: '' - password1: '[rac()ecar]' - password2: '74'
The username is invalid. It should have at least 1 character.
Test 3... Inputs:
username: 'sibel1' - password1: '[rac()ecar]' - password2: '74'
The username is invalid. It should have letters only.
Test 4... Inputs:
username: 'sibel' - password1: 'pass[]' - password2: '74'
The password1 is invalid. It should have at least 8 characters.
Test 5... Inputs:
username: 'sibel' - password1: 'abcdabcd' - password2: '74'
The password1 is invalid. It should have at least 2 brackets.
Test 6... Inputs:
username: 'sibel' - password1: '[[[[]]]]' - password2: '74'
The password1 is invalid. It should have letters too.
Test 7... Inputs:
username: 'sibel' - password1: '[no](no)' - password2: '74'
The password1 is invalid. It should have at least 1 character from the username.
Test 8... Inputs:
username: 'sibel' - password1: '[rac()ecar]]' - password2: '74'
The password1 is invalid. It should be balanced.
Test 9... Inputs:
username: 'sibel' - password1: '[rac()ecars]' - password2: '74'
The password1 is invalid. It should be possible to obtain a palindrome from the password1.
Test 10... Inputs:
username: 'sibel' - password1: '[rac()ecar]' - password2: '5'
The password2 is invalid. It should be between 10 and 10,000.

Test 11... Inputs:
username: 'sibel' - password1: '[rac()ecar]' - password2: '35'
The password2 is invalid. It is not compatible with the denominations.

6. RUNNING AND RESULTS



```
ionas@DESKTOP-U82RF4H:/mnt/c/Users/Yunus/OneDrive/Masaüstü/data/hw4$ javac *.java
ionas@DESKTOP-U82RF4H:/mnt/c/Users/Yunus/OneDrive/Masaüstü/data/hw4$ java TestClass
The username and passwords are valid. The door is opening, please wait..
ionas@DESKTOP-U82RF4H:/mnt/c/Users/Yunus/OneDrive/Masaüstü/data/hw4$ javac *.java
ionas@DESKTOP-U82RF4H:/mnt/c/Users/Yunus/OneDrive/Masaüstü/data/hw4$ java TestClass
The username is invalid due to username size
ionas@DESKTOP-U82RF4H:/mnt/c/Users/Yunus/OneDrive/Masaüstü/data/hw4$ javac *.java
ionas@DESKTOP-U82RF4H:/mnt/c/Users/Yunus/OneDrive/Masaüstü/data/hw4$ java TestClass
The username is invalid. It should have letters only
ionas@DESKTOP-U82RF4H:/mnt/c/Users/Yunus/OneDrive/Masaüstü/data/hw4$ javac *.java
ionas@DESKTOP-U82RF4H:/mnt/c/Users/Yunus/OneDrive/Masaüstü/data/hw4$ java TestClass
Password is invalid.password has to 8 charecter
ionas@DESKTOP-U82RF4H:/mnt/c/Users/Yunus/OneDrive/Masaüstü/data/hw4$ javac *.java
ionas@DESKTOP-U82RF4H:/mnt/c/Users/Yunus/OneDrive/Masaüstü/data/hw4$ java TestClass
The password1 is invalid. It should have at least 2 brackets.
ionas@DESKTOP-U82RF4H:/mnt/c/Users/Yunus/OneDrive/Masaüstü/data/hw4$ javac *.java
ionas@DESKTOP-U82RF4H:/mnt/c/Users/Yunus/OneDrive/Masaüstü/data/hw4$ java TestClass
Password is invalid. It has to includes at least one the same letter as username
ionas@DESKTOP-U82RF4H:/mnt/c/Users/Yunus/OneDrive/Masaüstü/data/hw4$ javac *.java
ionas@DESKTOP-U82RF4H:/mnt/c/Users/Yunus/OneDrive/Masaüstü/data/hw4$ java TestClass
Password is invalid. It has to includes at least one the same letter as username
ionas@DESKTOP-U82RF4H:/mnt/c/Users/Yunus/OneDrive/Masaüstü/data/hw4$ javac *.java
ionas@DESKTOP-U82RF4H:/mnt/c/Users/Yunus/OneDrive/Masaüstü/data/hw4$ java TestClass
The password1 is invalid. It should be balanced.
```



```
ionas@DESKTOP-U82RF4H:/mnt/c/Users/Yunus/OneDrive/Masaüstü/data/hw4$ java TestClass
The password1 is invalid. It should be possible to obtain a palindrome from the password1.
ionas@DESKTOP-U82RF4H:/mnt/c/Users/Yunus/OneDrive/Masaüstü/data/hw4$ javac *.java
ionas@DESKTOP-U82RF4H:/mnt/c/Users/Yunus/OneDrive/Masaüstü/data/hw4$ java TestClass
The password2 is invalid. It should be between 10 and 10,000.
ionas@DESKTOP-U82RF4H:/mnt/c/Users/Yunus/OneDrive/Masaüstü/data/hw4$ javac *.java
ionas@DESKTOP-U82RF4H:/mnt/c/Users/Yunus/OneDrive/Masaüstü/data/hw4$ java TestClass
The password2 is invalid. It is not compatible with the denominations.
ionas@DESKTOP-U82RF4H:/mnt/c/Users/Yunus/OneDrive/Masaüstü/data/hw4$
```

Only excepted output6 different then my output. My output says that at least you must have a letter that exist in username. In my opinion this is much better than excepted output.