CSE222

HomeWork #4

Report

1901042676

Mustafa Mercan

My approach to the problem

Step1: I created 4 different classes according to the data types and functionalities of the functions described in the homework pdf.

→ The first class I created is the user class, which will hold the username of the user. It also contains the checkIfValidUsername() method requested in the homework. In addition to this method, there are also other helper methods in the class.

→The second class I created is the password1 class, which will hold the username and a string password of the user. This class contains the part2 (containsUserNameSpirit()), part3 (isBalancedPassword()), and part4 (isPalindromePossible()) methods requested in the homework. There are also other helper methods in the class.

→The third class I created is the password2 class, which will hold an integer password of the user. This class contains the part5 (isExactDivision()) method requested in the homework.

→Finally, I created the checkPassword class to check the methods and parameters in the user, password1, and password2 objects created in the main.

Step2: After placing the necessary methods into the classes according to their functionalities, I started filling in the content of these methods. During this process, I also created the necessary helper methods in the relevant places.

Step3: After completing all of my methods and passing the tests I conducted separately for each class, I called the necessary methods for the controls inside the class I created at the end.

Step4: Finally, after testing the inputs given to us and some other inputs, I found that my application was running successfully and I ended my project here.

Test-Cases

Inputs:

```
// TEST 1
user user_1 = new user("sibelgulmez");
password1 password_1 = new password1("sibelgulmez","[rac()ecar]");
password2 password2_1 = new password2(74);
checkTest check_values_1 = new checkTest(user_1,password_1,password2_1);
check_values_1.check_values(new int[]{4,17,29});

// TEST 2
user user_2 = new user("");
password1 password_2 = new password1("","[rac()ecar]");
password2 password2_2 = new password2(74);
checkTest check_values_2 = new checkTest(user_2,password_2,password2_2);
check_values_2.check_values(new int[]{4,27,29});

// TEST 3
user user_3 = new user("sibel1");
password1 password_3 = new password1("sibel1","[rac()ecar]");
password2 password2_3 = new password2(74);
checkTest check_values_3 = new checkTest(user_3,password_3,password2_3);
check_values_3.check_values(new int[]{4,27,29});

//TEST 4
user user_4 = new user("sibel");
password1 password_4 = new password1("sibel","pass[]");
password2 password2_4 = new password2(74);
checkTest check_values_4 = new checkTest(user_4,password_4,password2_4);
check_values_4.check_values(new int[]{4,27,29});

//TEST 5
user user_5 = new user("sibel");
password1 password_5 = new password1("sibel","abcdabcd");
password2 password2_5 = new password2(74);
checkTest check_values_5 = new checkTest(user_5,password_5,password2_5);
check_values_5.check_values(new int[]{5,27,29});

//TEST 6
user user_6 = new user("sibel");
password1 password_6 = new password1("sibel","[[[[]]]]");
password2 password2_6 = new password2(32);
checkTest check_values_6 = new checkTest(user_6,password_6,password2_6);
check_values_6.check_values(new int[]{3,7,32});
```

```
    // TEST 7
    user user_7 = new user("sibel");
    password1 password_7 = new password1("sibel","[no](no)");
    password2 password2_7 = new password2(33);
    checkTest check_values_7 = new checkTest(user_7,password_7,password2_7);
    check_values_7.check_values(new int[]{3,7,32});

    // TEST 8
    user user_8 = new user("test");
    password1 password_8 = new password1("sibel","[rac()ecar]]");
    password2 password2_8 = new password2(79);
    checkTest check_values_8 = new checkTest(user_8,password_8,password2_8);
    check_values_8.check_values(new int[]{3,7,32});

    // TEST 9
    user user_9 = new user("test");
    password1 password_9 = new password1("sibel","[rac()ecars]");
    password2 password2_9 = new password2(78);
    checkTest check_values_9 = new checkTest(user_9,password_9,password2_9);
    check_values_9.check_values(new int[]{3,7,32});

    // TEST 10
    user user_10 = new user("test");
    password1 password_10 = new password1("sibel","[rac()ecar]");
    password2 password2_10 = new password2(5);
    checkTest check_values_10 = new checkTest(user_10,password_10,password2_10);
    check_values_10.check_values(new int[]{3,7,32});


    // TEST 11
    user user_11 = new user("test");
    password1 password_11 = new password1("sibel","[rac()ecar]");
    password2 password2_11 = new password2(35);
    checkTest check_values_11 = new checkTest(user_11,password_11,password2_11);
    check_values_11.check_values(new int[]{3,7,32});
}
```

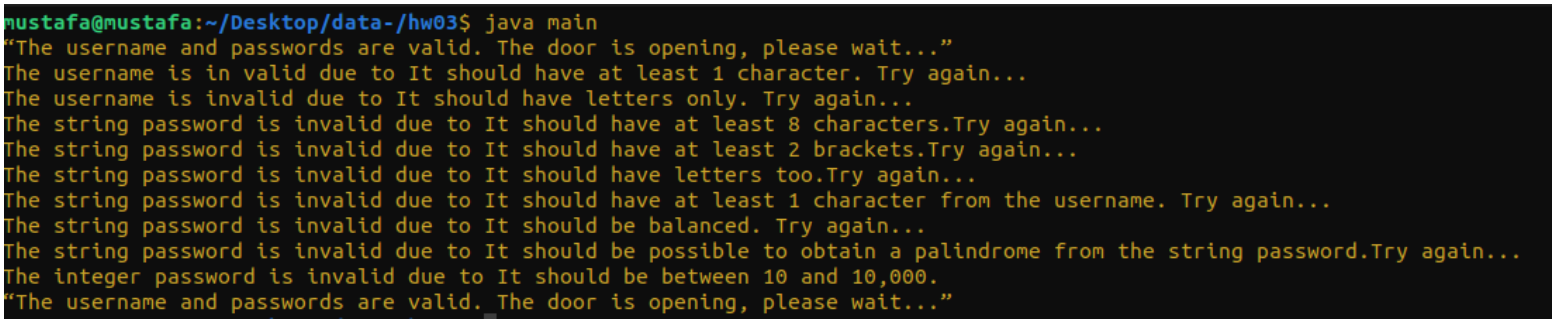
Content checkTest Class:

```
public class checkTest{
    public user user_1;
    public password1 password_1;
    public password2 password2_1;

    public checkTest(user user_1,password1 password_1, password2 password2_1){
        this.user_1 = user_1;
        this.password_1 = password_1;
        this.password2_1 = password2_1;
    }

    public void check_values(int [] denominations)
    {
        if(user_1.checkUserName() == true)
        {
            if(password_1.checkPassword1() == true)
            {
                if(password_1.containsUserNameSpirit() == true)
                {
                    if(password_1.isBalancedPassword() == true)
                    {
                        if(password_1.isPalindromePossible() == true)
                        {
                            if(password2_1.checkPassword2()== true)
                            {
                                if(password2_1.isExactDivision(denominations) == true)
                                {
                                    System.out.println(""The username and passwords are valid. The door is opening, please wait..."");
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

Outputs:



Time Complexity Analysis Of Each Function:

checkIfValidUsername(int currentIndex)
→The worst-case time complexity of the function is O(n) since it checks all the characters of the username array to determine whether they are alphabetical or not. Here, n represents the length of the userName array.

public boolean checkUserName()
→The function checks all characters of the username by calling the checkIfValidUsername function, which has a time complexity of O(n), where n is the length of the userName array. Therefore, the worst-case time complexity of the function is O(n).

containsUserNameSpirit()
→For this operation, all letters in both strings are processed once, so the time complexity of the function will be O(n). Here, n is the total number of letters in the user's password and username.

isBalancedPassword()
→This function checks the balancing of brackets in the user's password. The first loop, which processes all the brackets in the password once, has a time complexity of O(n). The second loop, which may need to check all the brackets again in the worst case, also has a time complexity of O(n). Therefore, the overall time complexity of the function is O(n). Here, n is the number of brackets in the user's password.

checkBracket(char ch)
→This function simply checks if a character is a bracket or not, so its time complexity is O(1).

check_match_bracket(char last_bracket,char first_bracket)
→This function checks whether two bracket characters match, so its time complexity is O(1).

public boolean isExactDivision(int num, int[] denominations, int idx, int remaining)
→This function is a recursive function that checks the exact divisibility of a number by using integer values within an array. The worst-case time complexity of this function depends on the first integer value in the array (denominations[0]) and the value of the number (num). Therefore, the time complexity will be O(b^m), where b is the first integer value in the array and m is the value of the number.

public boolean isExactDivision(int[] denominations)
→The worst-case time complexity of the isExactDivision() function depends on the first integer value in the array (denominations[0]) and the value of the number (num). Therefore, the worst-case time complexity of this function is also O(b^m), where b is the first integer value in the array and m is the value of the number. However, since this function is called only once, the total time complexity will still be O(b^m).