GTU Department of Computer Engineering

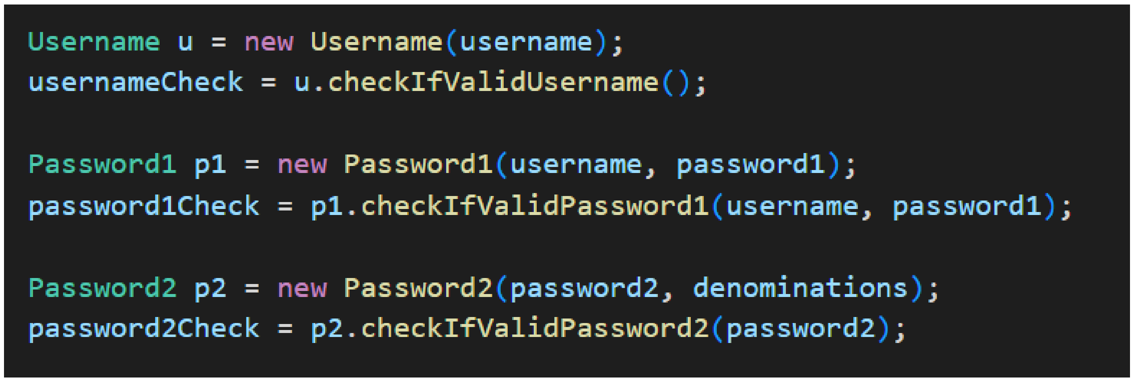
CSE 222/505 - Spring 2023

Homework 4 Report

BERRU LAFCI

1901042681

In my code there are 3 main method for checking whole conditions for username, password1 and password2. In main:

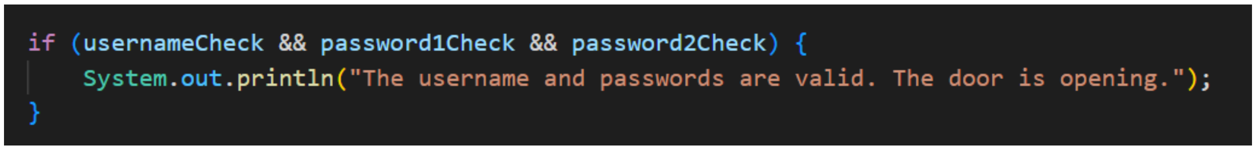


```java
Username u = new Username(username);
usernameCheck = u.checkIfValidUsername();

Password1 p1 = new Password1(username, password1);
password1Check = p1.checkIfValidPassword1(username, password1);

Password2 p2 = new Password2(password2, denominations);
password2Check = p2.checkIfValidPassword2(password2);
```

And I check the true for all condition in main also:



```java
if (usernameCheck && password1Check && password2Check) {
    System.out.println("The username and passwords are valid. The door is opening.");
}
```

checkIfValidUsername: It recursively travers the string if it contains only letters and has at least one character. So the complexity is O(n).

containsUserNameSpirit: The method uses a stack to push each character of the username, which takes O(n) time. Then, it uses a while loop to iterate through each character of the stack, which takes O(n) time as well. With the while loop, it uses another for loop to iterate through each character of the password1 to find same letter, which takes O(m) time.

So, the time complexity is O(n*m).

isBalancedPassword: The method uses a stack to push and pop each opening and closing bracket in the password1. It iterates through each character in the password1 using a for loop, which takes O(n) time. With the for loop, the method performs a constant amount of operations for each character, including pushing, popping, and comparing characters.

So, the time complexity is O(n).

isPalindromePossible: The complexity is O(n) when removing brackets from string. Then it goes to recursive function which is isPalindromePossibleHelper.

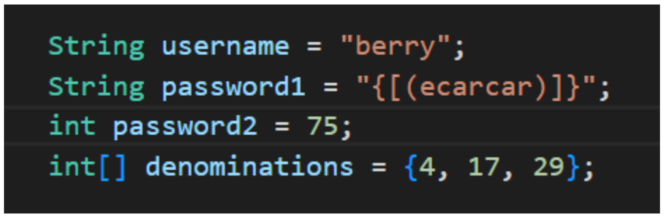
The method uses recursion to iterate through each character of the password1. For each character, it performs a constant number of operations, including checking if the character is contained in a string, adding or removing the character from the string, and making a recursive call with an incremented index. The time complexity of each operation is constant. However, the method uses the String replace() method, which has a worst-case time complexity of O(n) if it should traverse the entire string to replace the chracter.

So, the total time complexity is O(n^2) with also isPalindromePossible method. Because O(n^2) has faster growth than O(n).

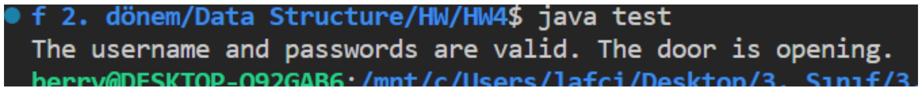
isExactDivision: The code iterates through the denominations array, trying to subtract each denomination from the password2 to see if it can be exactly divided by it. If it can, the function returns true, otherwise, it tries the next denomination. This process continues until all possible combinations of denominations have been tried or the password has been exactly divided.

The time complexity of this code is O(2^n), where n is the length of the denominations array. This is because the code uses recursion to explore all possible combinations of denominations to determine whether the password can be exactly divided by them. In the worst case scenario, the code will have to explore all possible combinations of denominations. So, the total time complexity is O(2^n).

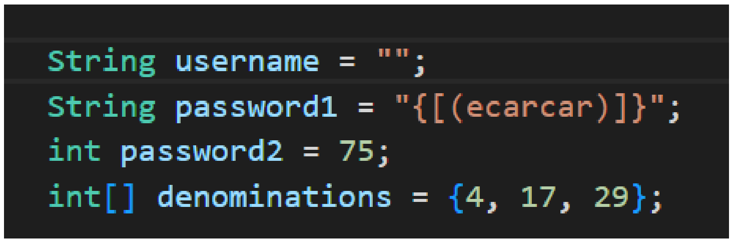
TEST CASES:



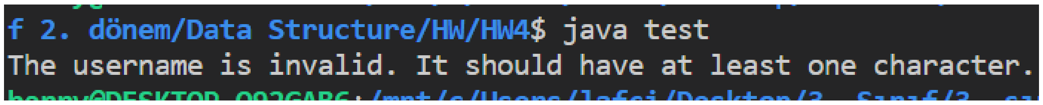
```
String username = "berry";
String password1 = "{[(ecarcar)]}";
int password2 = 75;
int[] denominations = {4, 17, 29};
```



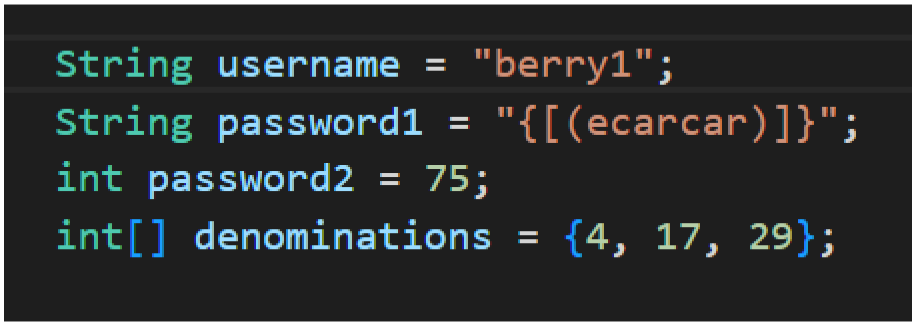
```
f 2. dönem/Data Structure/HW/HW4$ java test
The username and passwords are valid. The door is opening.
berry@DESKTOP-O92GAB6:/mnt/c/Users/lafci/Desktop/3. Sınıf/3.
```



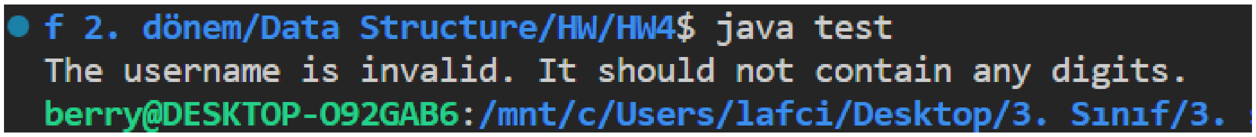
```
String username = "";
String password1 = "{[(ecarcar)]}";
int password2 = 75;
int[] denominations = {4, 17, 29};
```



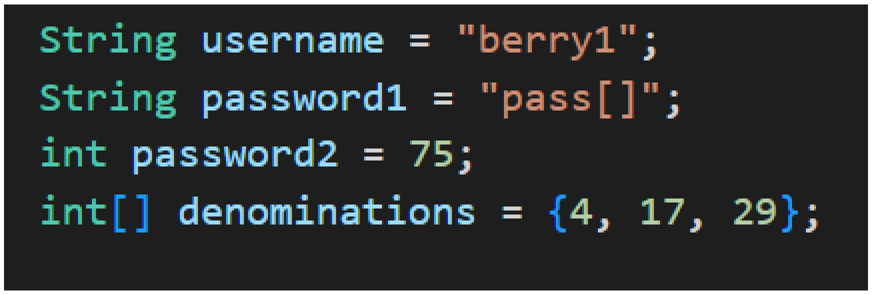
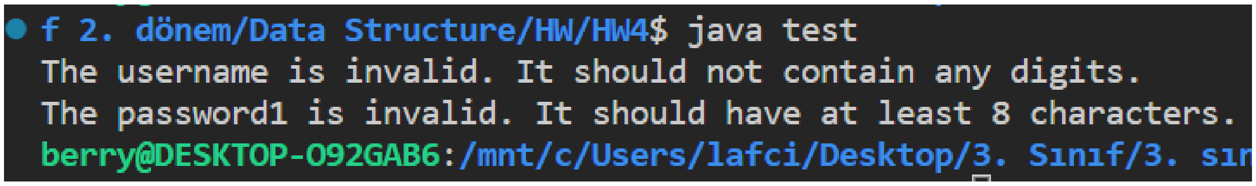
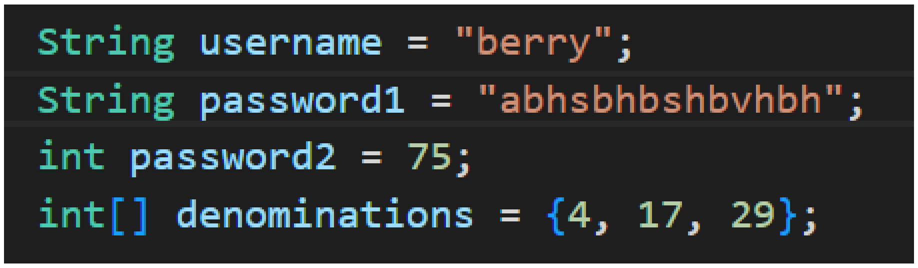
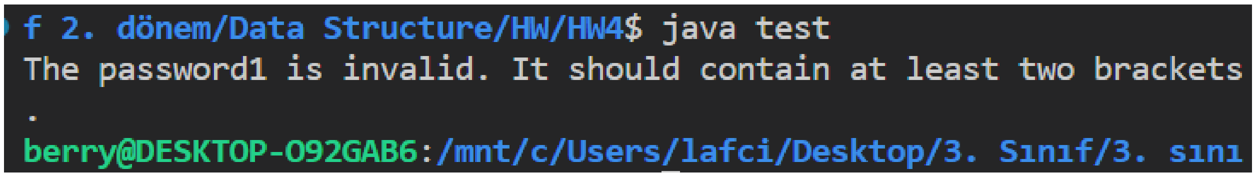
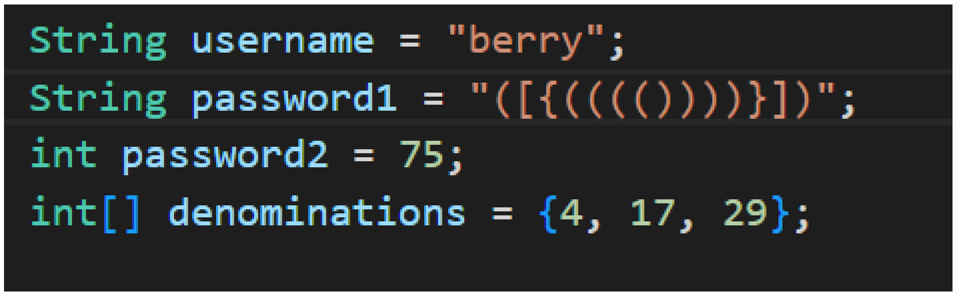
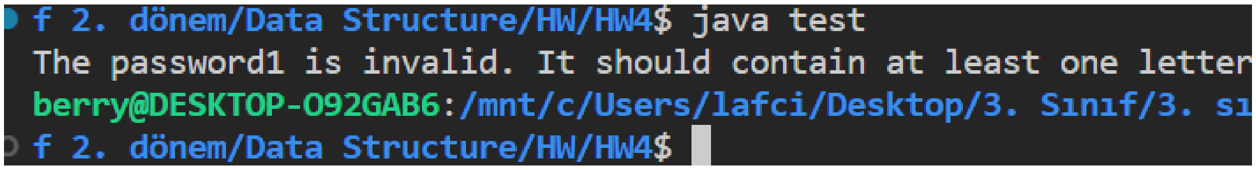
```
f 2. dönem/Data Structure/HW/HW4$ java test
The username is invalid. It should have at least one character.
berry@DESKTOP-O92GAB6:/mnt/c/Users/lafci/Desktop/3. Sınıf/3.
```

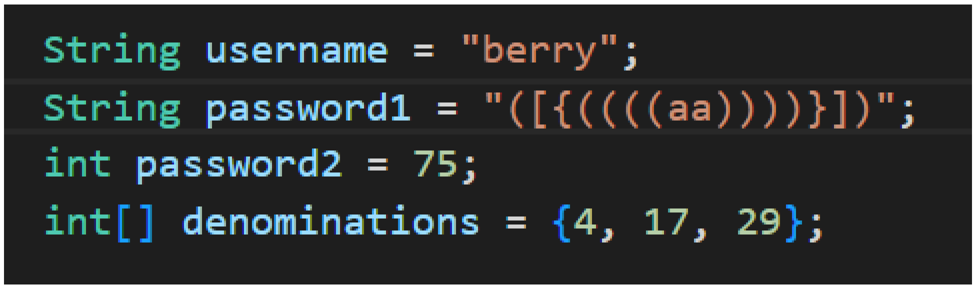


```
String username = "berry1";
String password1 = "{[(ecarcar)]}";
int password2 = 75;
int[] denominations = {4, 17, 29};
```

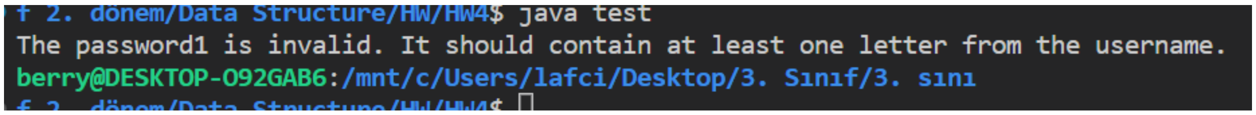


```
f 2. dönem/Data Structure/HW/HW4$ java test
The username is invalid. It should not contain any digits.
berry@DESKTOP-O92GAB6:/mnt/c/Users/lafci/Desktop/3. Sınıf/3.
```

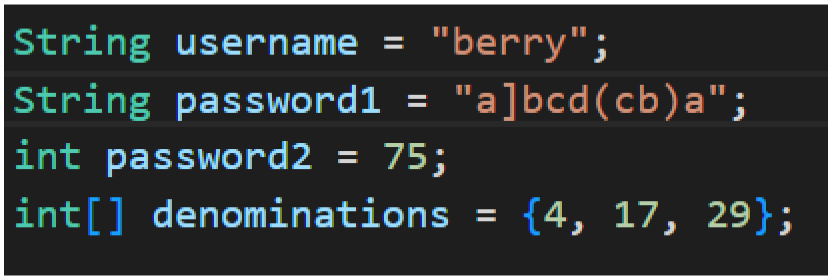

```
String username = "berry1";
String password1 = "pass[]";
int password2 = 75;
int[] denominations = {4, 17, 29};
```

```
f 2. dönem/Data Structure/HW/HW4$ java test
The username is invalid. It should not contain any digits.
The password1 is invalid. It should have at least 8 characters.
berry@DESKTOP-092GAB6:/mnt/c/Users/lafci/Desktop/3. Sınıf/3. sın
```

```
String username = "berry";
String password1 = "abhsbhbshbvhbh";
int password2 = 75;
int[] denominations = {4, 17, 29};
```

```
f 2. dönem/Data Structure/HW/HW4$ java test
The password1 is invalid. It should contain at least two brackets
.
berry@DESKTOP-092GAB6:/mnt/c/Users/lafci/Desktop/3. Sınıf/3. sını
```

```
String username = "berry";
String password1 = "([{(((()))}])";
int password2 = 75;
int[] denominations = {4, 17, 29};
```

```
f 2. dönem/Data Structure/HW/HW4$ java test
The password1 is invalid. It should contain at least one letter
berry@DESKTOP-092GAB6:/mnt/c/Users/lafci/Desktop/3. Sınıf/3. sı
f 2. dönem/Data Structure/HW/HW4$
```



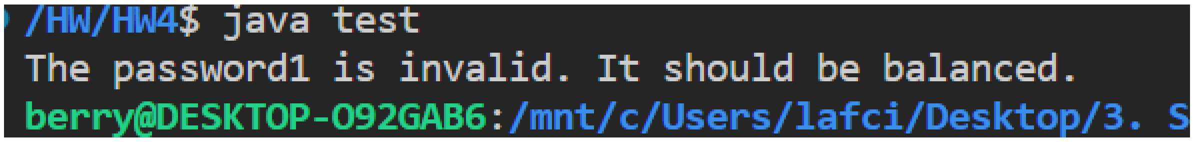
```
String username = "berry";
String password1 = "([{(((((aa)))))}])";
int password2 = 75;
int[] denominations = {4, 17, 29};
```



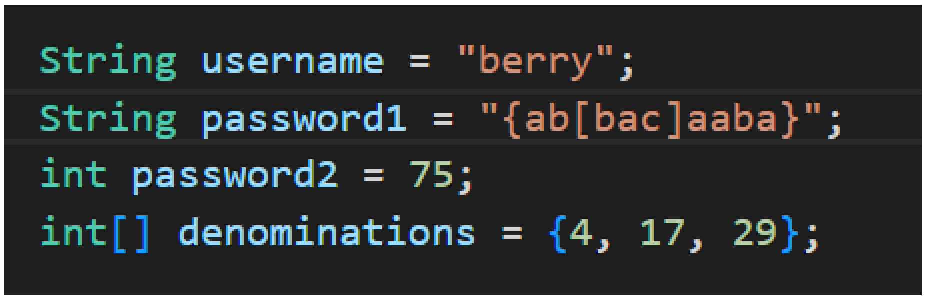
```
f 2. dönem/Data Structure/HW/HW4$ java test
The password1 is invalid. It should contain at least one letter from the username.
berry@DESKTOP-092GAB6:/mnt/c/Users/lafci/Desktop/3. Sınıf/3. sını
f 2. dönem/Data Structure/HW/HW4$
```



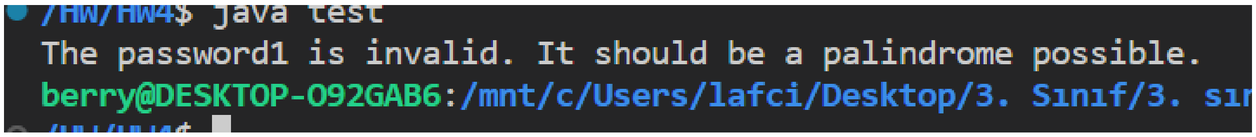
```
String username = "berry";
String password1 = "a]bcd(cb)a";
int password2 = 75;
int[] denominations = {4, 17, 29};
```



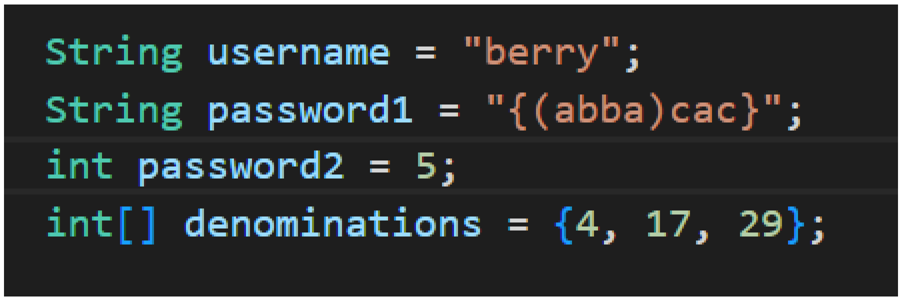
```
/HW/HW4$ java test
The password1 is invalid. It should be balanced.
berry@DESKTOP-092GAB6:/mnt/c/Users/lafci/Desktop/3. S
```



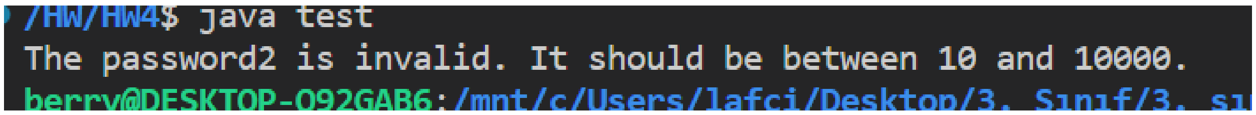
```
String username = "berry";
String password1 = "{ab[bac]aaba}";
int password2 = 75;
int[] denominations = {4, 17, 29};
```



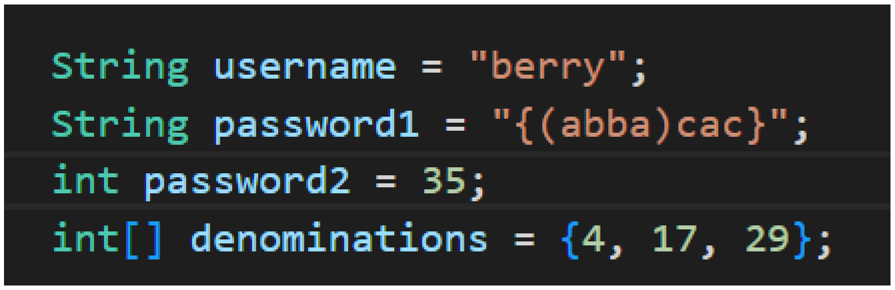
```
/HW/HW4$ java test
The password1 is invalid. It should be a palindrome possible.
berry@DESKTOP-092GAB6:/mnt/c/Users/lafci/Desktop/3. Sınıf/3. sı
/HW/HW4$
```



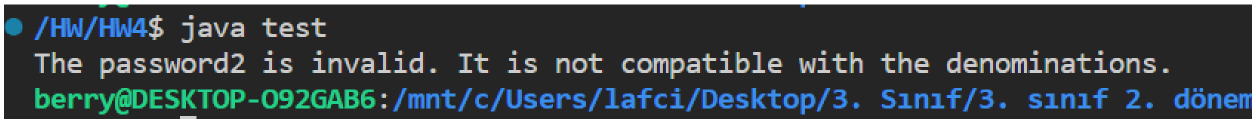
```java
String username = "berry";
String password1 = "{(abba)cac}";
int password2 = 5;
int[] denominations = {4, 17, 29};
```



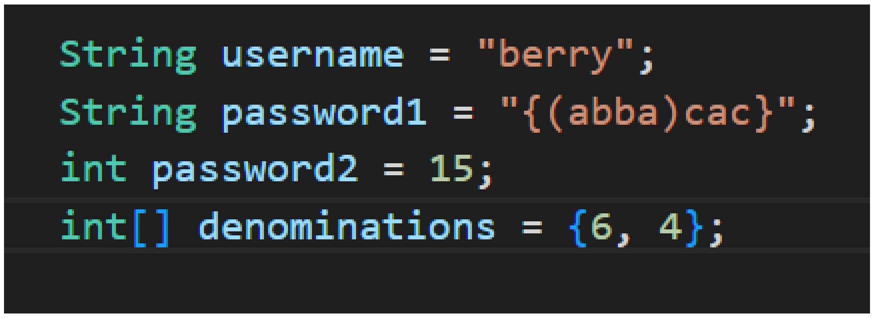
```
/HW/HW4$ java test
The password2 is invalid. It should be between 10 and 10000.
berry@DESKTOP-O92GAB6:/mnt/c/Users/lafci/Desktop/3. Sınıf/3. sı
```



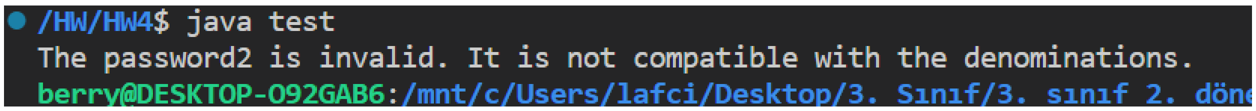
```java
String username = "berry";
String password1 = "{(abba)cac}";
int password2 = 35;
int[] denominations = {4, 17, 29};
```



```
/HW/HW4$ java test
The password2 is invalid. It is not compatible with the denominations.
berry@DESKTOP-O92GAB6:/mnt/c/Users/lafci/Desktop/3. Sınıf/3. sınıf 2. dönem
```



```java
String username = "berry";
String password1 = "{(abba)cac}";
int password2 = 15;
int[] denominations = {6, 4};
```



```
/HW/HW4$ java test
The password2 is invalid. It is not compatible with the denominations.
berry@DESKTOP-O92GAB6:/mnt/c/Users/lafci/Desktop/3. Sınıf/3. sınıf 2. döne
```