DAILY ONLINE ACTIVITIES SUMMARY

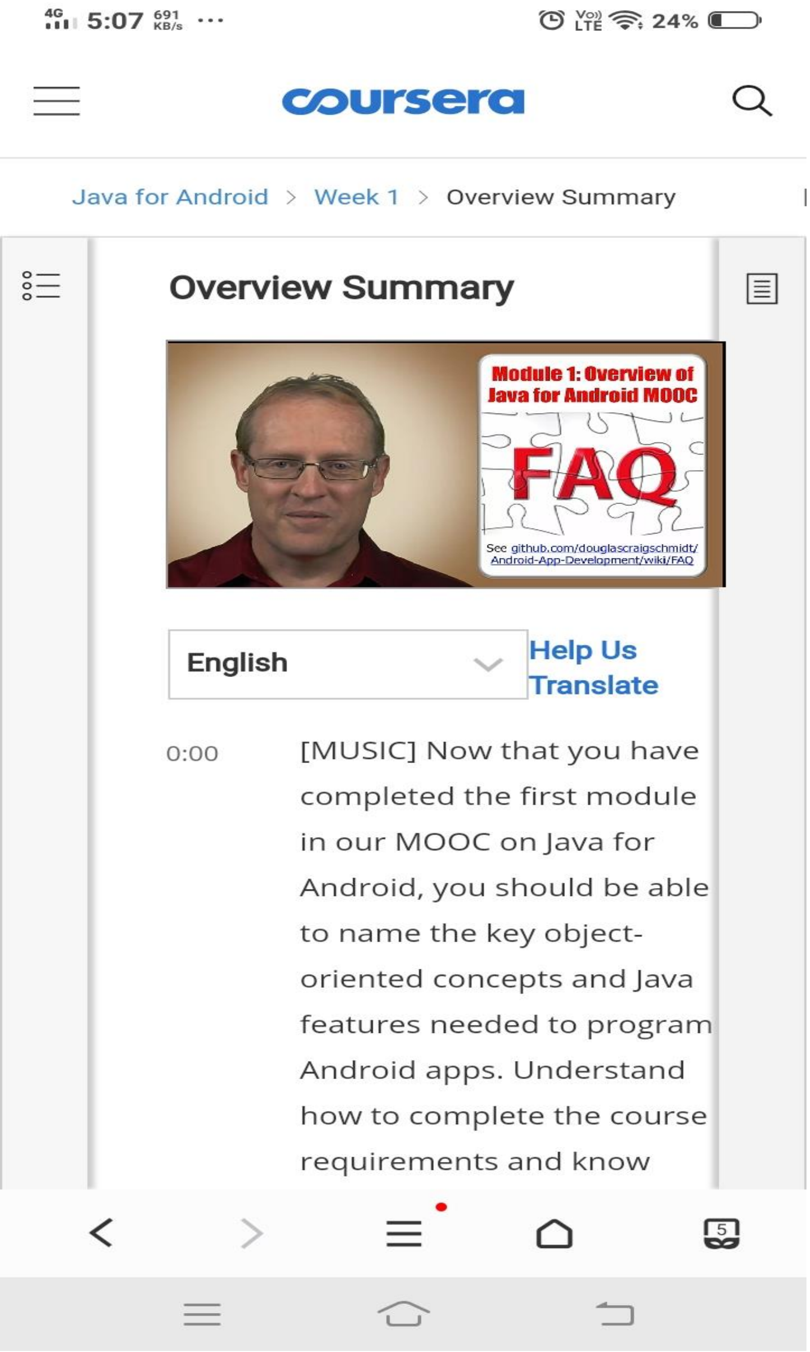
Date:	**14/07/2020**	**Name:**	**Prajwal**
Sem & Sec	**IV sem & B sec**	**USN:**	**4AL18CS057**
Online Test Summary			
Subject	**-----**		
Max. Marks	**-----**	**Score**	**------**
Certification Course Summary			
Course	**JAVA FOR ANDROID**		
Certificate Provider	**COURSEERA**	**Duration**	**4 WEEKS**
Coding Challenges			
Problem Statement: 1. Write a java program to check for balanced parenthesis.			
Status: Done			
Uploaded the report in Github	**YES**		
If yes Repository name	https://github.com/PRAJWALKOTIAN/lockdown-coding		
Uploaded the report in slack	**YES**		

Online test details

No test was conducted dated on 14 july 2020

Certification Course Details

The cource I have choosen is java for android in this I studied the overall summary of first week topics.

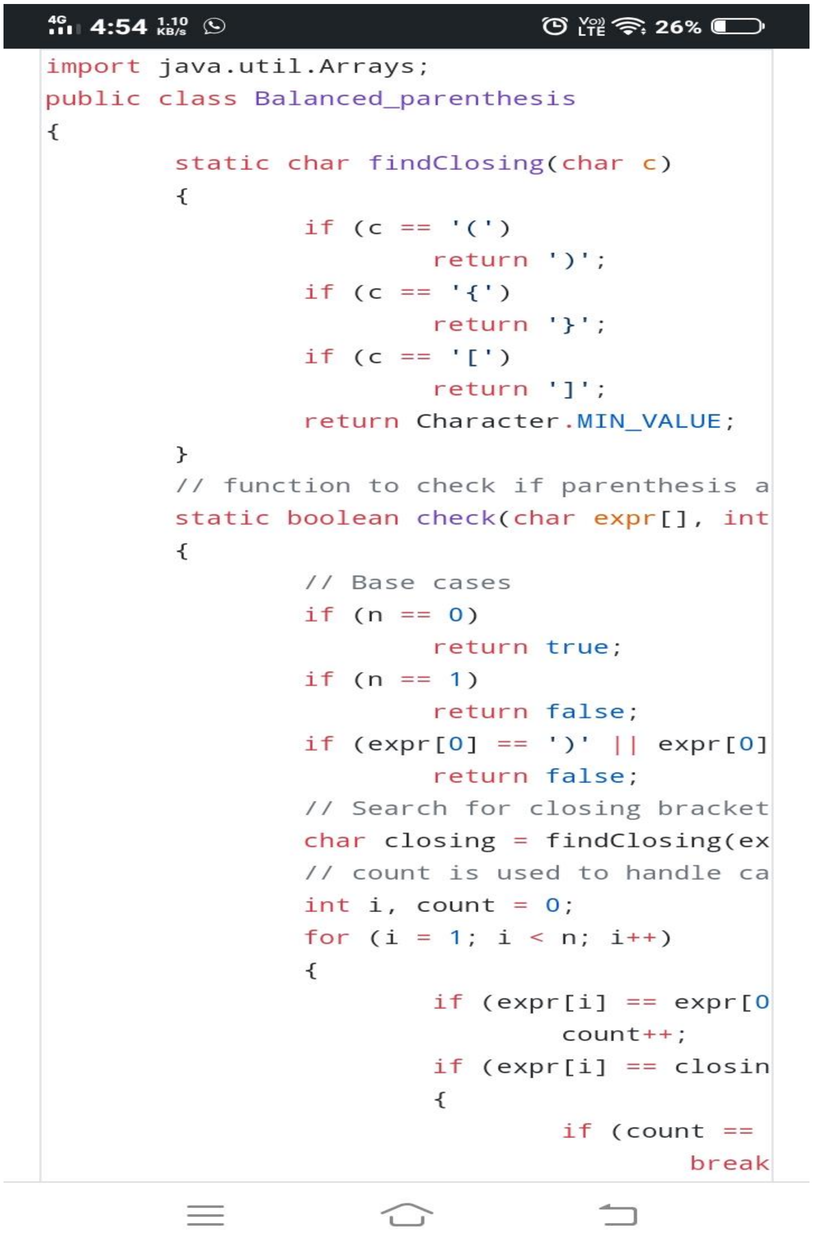

4G
5:07
691 KB/s
VoLTE
24%
coursera
Java for Android > Week 1 > Overview Summary
Overview Summary
Module 1: Overview of Java for Android MOOC
FAQ
See github.com/douglascraigschmidt/Android-App-Development/wiki/FAQ
English
Help Us Translate
5

0:00 [MUSIC] Now that you have completed the first module in our MOOC on Java for Android, you should be able to name the key object-oriented concepts and Java features needed to program Android apps. Understand how to complete the course requirements and know

Coding Challenges Details

The bellow given codes are there on my github repository https://github.com/PRAJWALKOTIAN/lockdown-coding

1. Write a java program to check for balanced parenthesis.



```
import java.util.Arrays;
public class Balanced_parenthesis
{
        static char findClosing(char c)
        {
                if (c == '(')
                        return ')';
                if (c == '{')
                        return '}';
                if (c == '[')
                        return ']';
                return Character.MIN_VALUE;
        }
        // function to check if parenthesis a
        static boolean check(char expr[], int
        {
                // Base cases
                if (n == 0)
                        return true;
                if (n == 1)
                        return false;
                if (expr[0] == ')' || expr[0]
                        return false;
                // Search for closing bracket
                char closing = findClosing(ex
                // count is used to handle ca
                int i, count = 0;
                for (i = 1; i < n; i++)
                {
                        if (expr[i] == expr[0
                                count++;
                        if (expr[i] == closin
                        {
                                if (count ==
                                        break
```