**Problem Name:** **Cultural Fest Seating Plan**

**Topic:** Algorithms

**Tags:** Dynamic Programming, Recursion, Arrays

**Language used: Java**

**Problem Statement:**

NIT Srinagar is hosting its annual Cultural Fest, and the organizers need to arrange seats for the event. The seats are divided into sections, and each section has a specific number of seats. The goal is to arrange the seats such that the number of VIP guests and regular guests in each section is as balanced as possible. If it's not possible to achieve a balanced arrangement, the organizers will have to declare the arrangement "not possible".

**Input Format:**

1. The first line of input contains an integer ‘n’ :- the number of sections.
2. The second line contains n space-separated integers representing the number of seats in each section.

**Output Format:**

1. Output a single string: "balanced" if the seating arrangement can be balanced or "not possible" if it cannot.

**Constraints:**

* 1 <= n <= 100
* 1 <= seats[i] <= 1000

**Sample Case 1:**

**Input:**

4

10 20 15 25

**Output:**

balanced

**Explanation for Sample case 1:**

**Step 1: Initialization**

* Calculate the total number of seats:



* Since totalSeats is even, proceed with the next steps.Calculate the target seats for balancing:



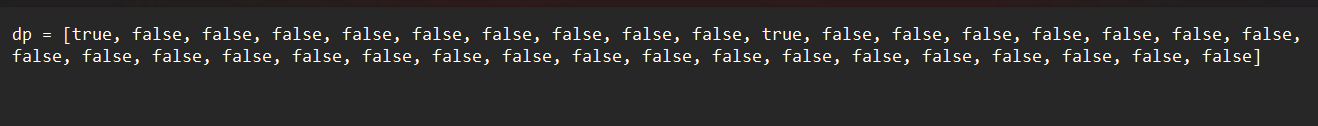
* Initialize a boolean array dp of size targetSeats + 1 with dp[0] set to true
* dp = [true, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false]

**Step 2: Dynamic Programming**

Iterate through the sections and update the dp array.

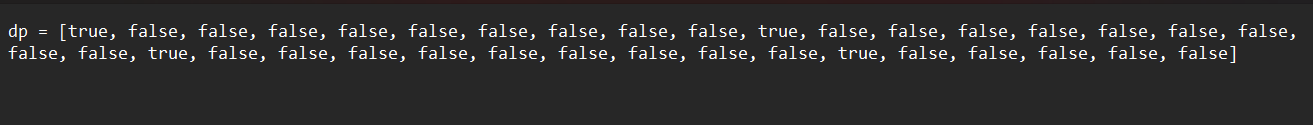
1. Section: 10

Update dp[10] to true:



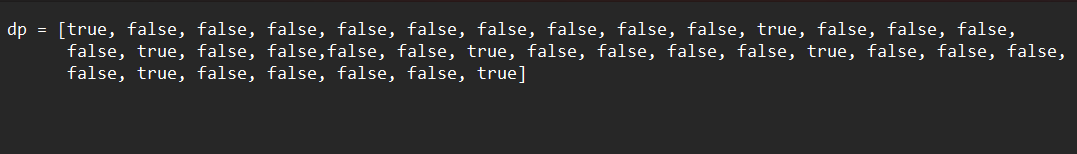
1. Section: 20

Update dp[20] and dp[30] to true:



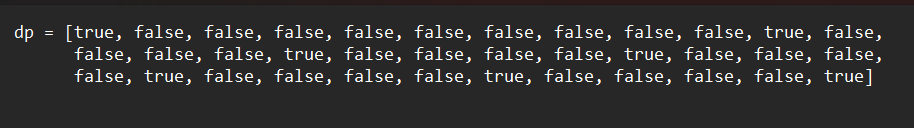
1. Section: 15

Update dp[15], dp[25], and dp[35] to true:



1. Section: 25

Update dp[25] and dp[35] (already true):



**Step 3: Conclusion**

* Since dp[targetSeats] (i.e., dp[35]) is true, we conclude that the seats can be balanced between VIP and regular guests.
* Output: ‘balanced’

**Sample Case 2:**

Sample Input:



Sample Output:



**Explanation:**

* Step 1: Initialization
* Calculate the total number of seats:



* Since totalSeats is odd, it is immediately clear that the seats cannot be divided evenly.
* Output: not possible

**Detailed Sample Case 3:**

Sample Input:



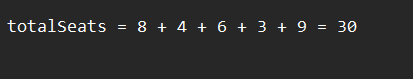
Sample Output:



**Explanation:**

**Step 1: Initialization**

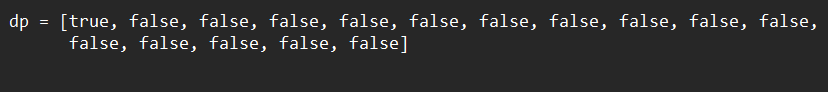
* Calculate the total number of seats



* Since totalSeats is even, proceed with the next steps.
* Calculate the target seats for balancing:



* Initialize a boolean array dp of size targetSeats + 1 with dp[0] set to true:

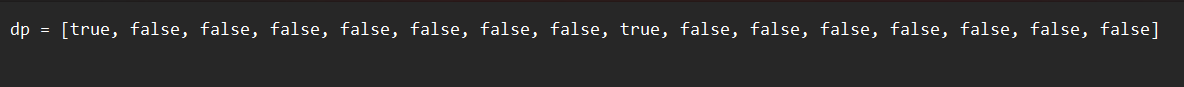


**Step 2: Dynamic Programming**

Iterate through the sections and update the dp array.

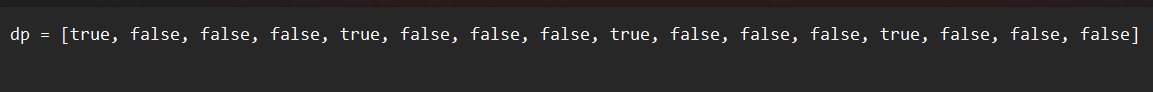
1. Section: 8

Update dp[8] to true:



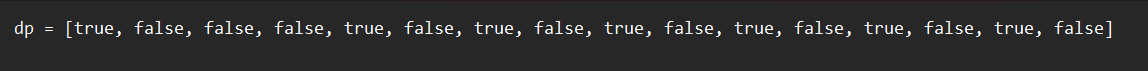
1. Section: 4

Update dp[4] and dp[12] to true:



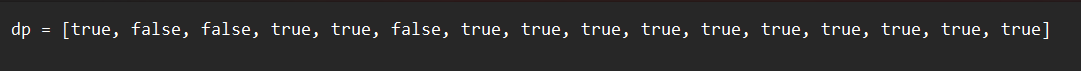
1. Section: 6

Update dp[6], dp[10], and dp[14] to true:



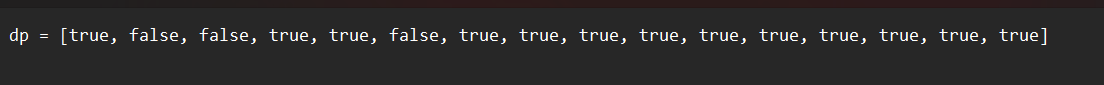
1. Section: 3

Update dp[3], dp[7], dp[9], dp[11], dp [13], and dp[15] to true:



1. Section: 9

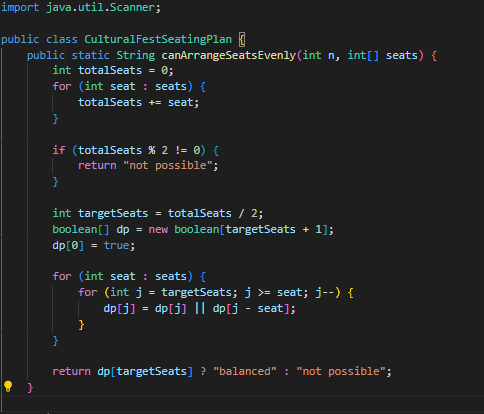
Update dp[9] and dp[15] (already true):

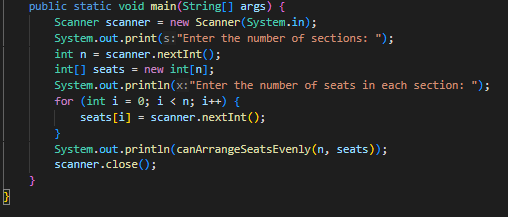


Step 3: ConclusionSince dp[targetSeats] (i.e., dp[15]) is true, we conclude that the seats can be balanced between VIP and regular guests.

Output: ‘balanced’

**Code:**





Notes: Save this Code as “ CulturalFestSeatingPlan.java “ .

**One Compiler Link:**

* [Cultural Fest Seating Plan](https://onecompiler.com/java/42dgtkmq9)