class SubsetSum {

static boolean canPartition(int[] num, int sum) {

//TODO: Write - Your - Code

return false;

}

}

SOLUTION:

class SubsetSum {

static boolean canPartition(int[] num, int sum) {

int n = num.length;

boolean[] dp = new boolean[sum + 1];

// handle sum=0, as we can always have '0' sum with an empty set

dp[0] = true;

// with only one number, we can have a subset only when the required sum is equal to its value

for (int s = 1; s <= sum; s++) {

dp[s] = (num[0] == s ? true : false);

}

// process all subsets for all sums

for (int i = 1; i < n; i++) {

for (int s = sum; s >= 0; s--) {

// if dp[s]==true, this means we can get the sum 's' without num[i], hence we can move on to

// the next number else we can include num[i] and see if we can find a subset to get the

// remaining sum

if (!dp[s] && s >= num[i]) {

dp[s] = dp[s - num[i]];

}

}

}

return dp[sum];

}

}