```java
/*Author: Bochen (mddboc@foxmail.com)
Last Modified: Tue Apr 10 22:28:44 CST 2018*/

/*Given an array S of n integers, find three integers in S such that the sum is closest to a given number, target. Return the sum of the three integers. You may assume that each input would have exactly one solution.

        For example, given array S = {-1 2 1 -4}, and target = 1.

        The sum that is closest to the target is 2. (-1 + 2 + 1 = 2).*/

import java.util.Arrays;
import java.lang.Math;
import java.lang.System;


public class Main {

    public static void main(String[] args)
    {
        int[] nums = {-10,0,-2,3,-8,1,-10,8,-8,6,-7,0,-7,2,2,-5,-8,1,-4,6};

        Solution solution = new Solution();
        int receive = solution.threeSumClosest(nums, 18);


        System.out.println("haha");
    }

}


class Solution {
    public int threeSumClosest(int[] nums, int target) {


        if (nums == null || nums.length < 3) {
            return 0;
        }

        Arrays.sort(nums);

        int numsLength = nums.length;
        int minValue = nums[0] + nums[1] + nums[2];
        if (minValue >= target) {
            return minValue;
        }
        int maxValue = nums[numsLength - 1] + nums[numsLength - 2] + nums[numsLength - 3];
        if (maxValue <= target) {
            return maxValue;
        }

        int startPointer = 0, endPointer = 0;
        int returnValue = minValue;
        int currentValue = 0;
        for (int i = 0; i < numsLength - 2; i++) {

            startPointer = i + 1;
            endPointer = numsLength - 1;

            while (startPointer < endPointer) {

                if ( startPointer != i+1 && nums[startPointer] == nums[startPointer-1]){
                    startPointer++;
                    continue;
                }
                if ( endPointer != numsLength-1 && nums[endPointer] == nums[endPointer+1]){
                    endPointer--;
                    continue;
                }
```

```
69
70                  currentValue = nums[i] + nums[startPointer] + nums[endPointer];
71                  if (currentValue > target) {
72                      endPointer--;
73                  } else if (currentValue < target) {
74                      startPointer++;
75                  } else {
76                      return target;
77                  }
78
79                  returnValue = Math.abs(currentValue - target) < Math.abs(returnValue
                    - target) ? currentValue : returnValue;
80              }
81          }
82
83          return returnValue;
84      }
85
86
87  }
```