|  |
| --- |
| package Test;  import java.io.BufferedReader;  import java.io.IOException;  import java.io.InputStreamReader;  import java.util.HashMap;  import java.util.Map;  class Solution{  public int[] twoSum(int[] nums,int target) {  Map<Integer,Integer> map = new HashMap<>();  for(int i=0;i<nums.length;i++) {  int complement = target - nums[i];  if(map.containsKey(complement)) {  return new int[] {map.get(complement),i};  }  map.put(nums[i], i);  }  throw new IllegalArgumentException("No two sum solution!");  }  }  public class TwoSum {  public static int[] stringToIntegerArray(String input) {  input = input.trim();  input = input.substring(1,input.length()-1);  if(input.length() == 0) {  return new int[0];  }  String[] parts = input.split(",");  int []output = new int[parts.length];  for(int index = 0;index<parts.length;index++) {  String part = parts[index].trim();  output[index] = Integer.parseInt(part);  }  return output;  }  public static String integerArrayToString(int[] nums,int length) {  if(length == 0) {  return "[]";  }  String result = "";  for(int index = 0;index < length;index++) {  int number = nums[index];  result += Integer.toString(number)+",";  }  return "["+result.substring(0,result.length()-2)+"]";  }  public static String integerArrayToString(int[] nums){  return integerArrayToString(nums,nums.length);  }  public static void main(String[] args) throws IOException{  BufferedReader in = new BufferedReader(new InputStreamReader(System.in));  String line;  while ((line = in.readLine())!= null) {  int []nums = stringToIntegerArray(line);  line = in.readLine();  int target = Integer.parseInt(line);  int[] ret = new Solution().twoSum(nums, target);  String out = integerArrayToString(ret);  System.out.print(out);  }  }  } |

BufferedReader/BufferedWriter是修饰类，提供缓冲，以及按行读写功能。

HashMap其中Map表示映射关系，是一个接口，实现Map接口有多种方式，HashMap实现的方式利用了哈希；Entry是一个内部类

HashMap的基本原理：内部有一个哈希表，即数组table，每个元素table[i]指向一个单向链表，根据键存取值，用键算出hash值，取模得到数组中的索引位置buketIndex，然后操作table[buketIndex]指向的单向链表。

存取的时候根据键的hash值，只在对应的链表中操作，不会访问别的链表，在对应的链表操作时也是先比较hash值，如果相同再用equals方法比较。这就要求，相同的对象其hashCode返回值必须相同，如果键是自定义的类，就特别需要注意这一点。这也是hashCode和equals方法的一个关键约束。

Java8对HashMap的实现进行了优化，在哈希冲突严重的情况下，会将该链表转换为一个平衡的排序二叉树。

Integer.parseInt(String)的作用就是将String字符类型数据转换为Integer整型数据