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

/*Say you have an array for which the ith element is the price of a given stock on day i.

        If you were only permitted to complete at most one transaction (ie, buy one and sell one share of the stock), design an algorithm to find the maximum profit.

        Example 1:
        Input: [7, 1, 5, 3, 6, 4]
        Output: 5

        max. difference = 6-1 = 5 (not 7-1 = 6, as selling price needs to be larger than buying price)
        Example 2:
        Input: [7, 6, 4, 3, 1]
        Output: 0

        In this case, no transaction is done, i.e. max profit = 0.*/

import java.util.*;
import java.lang.Math;
import java.lang.System;
import java.lang.Integer;


public class Main {

    public static void main(String[] args) throws ArithmeticException {

        int[] input = {7,1,5,3,6,4};

        Solution solution = new Solution();

        int result = solution.maxProfit(input);

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

}


class ListNode {
    int val;
    ListNode next;

    ListNode(int x) {
        val = x;
    }
}


class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;

    TreeNode(int x) {
        val = x;
    }
}


class Solution {
    public int maxProfit(int[] prices) {

        if (prices == null || prices.length < 2) {
            return 0;
        }

        int pricesLength = prices.length;
```

```
70
71          int result = 0;
72          int minPrice = prices[0];
73
74          for (int i = 1; i < pricesLength; i++) {
75
76              if (prices[i] > minPrice) {
77                  result = Math.max(result, prices[i] - minPrice);
78              } else {
79                  minPrice = prices[i];
80              }
81
82          }
83
84          return result;
85      }
86  }
```