Challenge 2- Stock price

Write a function to find out the best Buying and Selling day for maximum gain from daily stock prices of last 10 days. Following are 2 rules,

1. Buy first
2. You can only buy once and sell once

Example & Expected Output

Below table represents stock price for last 10 days:

Function should return Day 6 as buying day and Day 9 as selling day. Total gain is $53.

Day	Stock Price
Day 1	$3
Day 2	$8
Day 3	$8
Day 4	$55
Day 5	$38
Day 6	$1
Day 7	$7
Day 8	$42
Day 9	$54
Day 10	$53

Solution (Java)

The maximum gain will be the difference between best selling day and best buying day, i.e.

Maximum Gain = Best Selling Day Price – Best Buying Day Price

To find maximum gain, best buying day, best selling day

Basic Idea: Keeping in mind that buying day should come before the selling day, to find the maximum gain for a given day 'i', we need to find the highest price among the day's 'i' to 10th day/ last day and then subtract it with the price at day 'i'.

<u>Algorithm</u>:

1. Start from tenth day (i.e. i=10) and find the maximum price among the days 'i' to tenth day/last day and store maximum price obtained in 'maxValue' variable. Also, calculate the maximum possible gain if buying day is 10th day, by subtracting price on day 10 from maxValue and store the obtained gain in maximumGain.

   ➢ Since there are no days after 10th day and 10th is the last day,
     maxValue = price_10thDay = 53
     maximumGain = maxValue - price_10thDay = 53-53 = 0

2. Now, consider day 9 (i.e. i=9) and find the maximum price among days 'i' to tenth day/last day and again store maximum price in 'maxValue' variable. Again, calculate the maximum possible gain if buying day is 9th day, by subtracting price on day 9 from maxValue and store the obtained gain in maximumGain.

   ➢ maxValue = max(price_10thDay, price_9thDay) = max(53,54) = 54
     maximumGain = maxValue - price_9thDay = 54 – 54 = 0

3. Now, consider day 8 (i.e. i=8) and find the maximum price among days 'i' to tenth day/last day and again store maximum price in 'maxValue' variable. Again, calculate the maximum possible gain if buying day is 8th day, by subtracting price on day 8 from maxValue and store the obtained gain in maximumGain.

   ➢ maxValue = max(price_10thDay, price_9thDay, price_8thDay) = max(53, 54, 42) = 54
     maximumGain = maxValue - price_8thDay = 54 – 42 =12

4. Now, consider day 7 (i.e. i=7) and follow above steps and calculate maxValue and maximumGain.

   ➢ maxValue = max(price_10thDay, price_9thDay, price_8thDay, price_7thDay) = max(53, 54, 42, 7) = 54
     maximumGain = maxValue - price_7thDay = 54 – 7 =47

5. Now, consider day 6 (i.e. i=6) and follow above steps and calculate maxValue and maximumGain.

   ➢ maxValue = max(price_10thDay, price_9thDay, price_8thDay, price_7thDay, price_6thday) = max(53, 54, 42, 7, 1) = 54
     maximumGain = maxValue - price_6thDay = 54 – 1 = 53

6. Now, consider day 5 (i.e. i=5) and follow above steps and calculate maxValue and maximumGain.

   ➢ maxValue = max(53, 54, 42, 7, 1, 38) = 54
     maximumGain = maxValue - price_5thDay = 54 – 38 = 16

7. Now, consider day 4 (i.e. i=4) and follow above steps and calculate maxValue and maximumGain.

> maxValue = max(53,54,42,7,1, 38, 55) = 55
maximumGain = maxValue - price_4thDay = 55 – 55 = 0

8. Now, consider day 3 (i.e. i=3) and follow above steps and calculate maxValue and maximumGain.

> maxValue = max(53,54,42,7,1, 38, 55, 8) = 55
maximumGain = maxValue - price_3rdDay = 55 – 8 = 47

9. Now, consider day 2 (i.e. i=2) and follow above steps and calculate maxValue and maximumGain.

> maxValue = max(53,54,42,7,1, 38, 55, 8, 8) = 55
maximumGain = maxValue - price_2ndDay = 55 – 8 = 47

10. Now, consider day 1 (i.e. i=1) and follow above steps and calculate maxValue and maximumGain.

> maxValue = max(53,54,42,7,1, 38, 55, 8, 8, 3) = 55
maximumGain = maxValue - price_1stDay = 55 – 3 = 52

From the above maximumGain values, it is clear that maximum profit is **53** and the maximum profit is achieved when buying day is **day 6** and selling day is **day 9**. This algorithm will still be O(n) as we are scanning through the array only once.

Code
The simple function that could give the maximum profit is
```java
public int maxProfit(int[] prices) {
            int maxGain = 0;
            int maxValue = -1;
             int i= prices.length -1;

                while((i>=0)){
                if(prices[i] >= 0 && prices[i] > maxValue) {
                        maxValue = prices[i];
                        }

                if(prices[i] >= 0 && (maxValue - prices[i]) >= maxGain){
                    maxGain = maxValue - prices[i];
                }

                    i--;
```

```
        }
            return maxGain;
        }
```

This code works perfectly fine, but the only modification needed is to retrieve the best selling day and best buying day for which maximum gain is obtained along with maximum profit value. To achieve that, I implemented 'map' to store values of selling day, buying day, and maximum profit as shown below.

```java
public static Map<String, Integer> maxProfit(int[] prices) {

        int maxGain = 0;
        int maxValue = -1;
        int i= prices.length -1  ;
        int sellingDay = 0;
        int buyingDay = 0;

        Map<String, Integer> res = new HashMap<>();

        while((i>=0)){
        if(prices[i] >= 0 && prices[i] > maxValue) {
                    maxValue = prices[i];
                    sellingDay = i+1;
        }

        if(prices[i] >= 0 && (maxValue - prices[i]) >= maxGain){
                    maxGain = maxValue - prices[i];
                    buyingDay = i+1;
                    res.put("Buying Day", buyingDay);
                    res.put("Selling Day", sellingDay);
                    res.put("Maximum Gain", maxGain);

            }

        i--;

    }
    return res;
    }
```
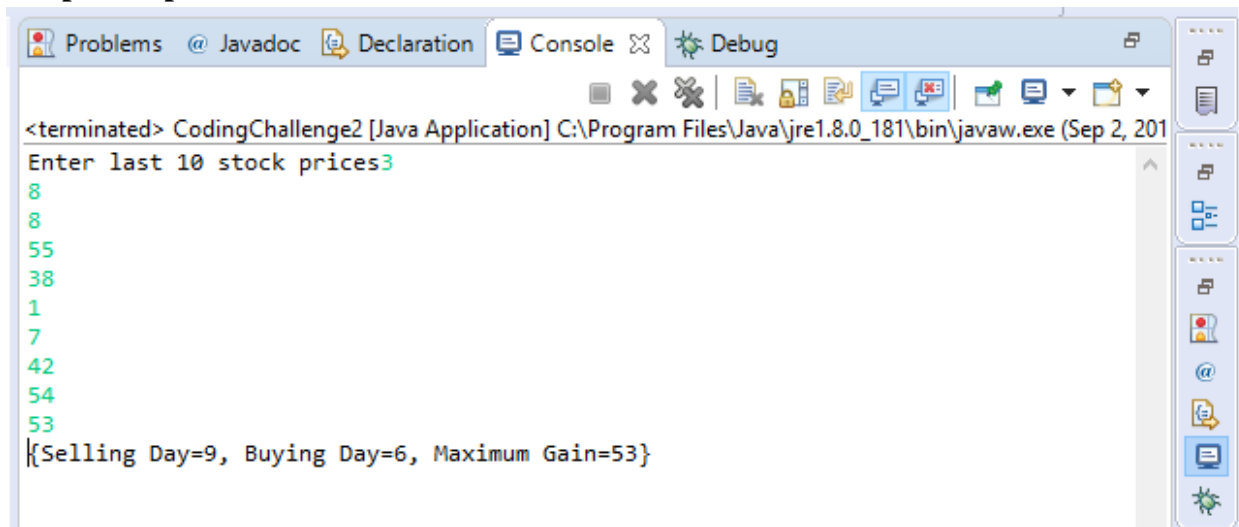
To download complete code, please find **CodingChallenge2.java** class in the same repository.

Sample Output


```
Problems  @ Javadoc  Declaration  Console ⌗  Debug

<terminated> CodingChallenge2 [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (Sep 2, 201
Enter last 10 stock prices3
8
8
55
38
1
7
42
54
53
{Selling Day=9, Buying Day=6, Maximum Gain=53}
```

Test Cases & Instructions to execute

In the console, as shown in the "Sample Output" above, please enter last 10 stock price values by hitting Enter for each value and after entering 10 values, please hit Enter again to see the output as below

Input: If last 10 stock price values are [3, 8, 8, 55, 38, 1, 7, 42, 54, 53]
Output: {Selling Day=9, Buying Day=6, Maximum Gain=53}

Negative entries: Even if one of the entries of prices is negative, the code will not get affected by it and shows the best buying, selling days and maximum profit correctly. For example, if the last 10 stock price values are [1, -1, 2, 3, 4, 5, 6, 7, 8, 8], then the output will be
{Selling Day=10, Buying Day=1, Maximum Gain=7}

In case of duplicate prices: If two are more days have same stock prices and if that particular stock price is a part of maximum gain either as selling price or as a buying price, then also we will achieve maximum gain by considering one of those selling days or buying days. For example, if the last 10 stock price values are [1, 2, 3, 4, 5, 6, 7, 8, 9, 9], then the output is
{Selling Day=10, Buying Day=1, Maximum Gain=7}
Here the value of day 9, day 10 are both 9 (which is the best selling price in this case), so the output here is showing day 10(one of them) as best selling day since the challenge is to find one best selling day & buying day that gives maximum profit.

Prices in descending order:
For example, if the last 10 stock price values are [9, 8, 7, 6, 5, 4, 3, 2, 1, 1], then the output will be
{Selling Day=1, Buying Day=1, Maximum Gain=0}
Maximum gain is zero here because prices kept going down and there is not profit.
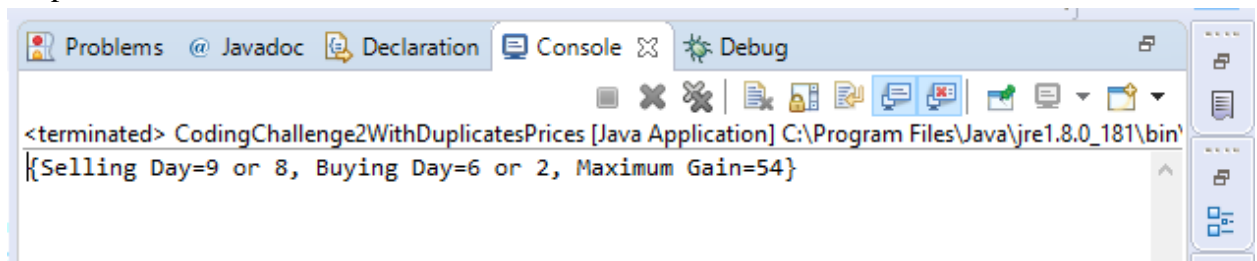
Extension
As an extension, I'm implementing one small feature which is- if the prices of two or more days are same and if prices of such days are either best selling/ best buying prices, then this program will give the list of all possible buying and selling days that gives the maximum gain.

Sample Output
> Input: If the prices of last 10 days are [3, 1, 8, 8, 2, 1, 7, 55, 55, 34]
> Output:


```
Problems  @ Javadoc  Declaration  Console ⊠  Debug
<terminated> CodingChallenge2WithDuplicatesPrices [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\
{Selling Day=9 or 8, Buying Day=6 or 2, Maximum Gain=54}
```

Code
The complete code to obtain the output as shown in above "Sample Output" (list of all possible buying and selling days that gives the maximum gain).

```java
package com.coding.challenge;

import java.util.HashMap;
import java.util.Map;

/**
 * CodingChallenge2WithDuplicatePrices.java
 * Purpose: To find out the best Buying and Selling days from daily stock
prices of last 10 days to get maximum profit.
 *
 * @author Nikhita
 *
 */
public class CodingChallenge2WithDuplicatesPrices {

    /**
```

```
        * This function takes the stock_prices of last ten days as an input.
        * Returns a Map which contains the information about the maximum
profit, best selling day and best buying day to get that maximum profit.
        * If best buying day and selling day prices are present more than once
in the table, then the function will return all best selling & buying days
        *
        * @param prices
        * @return Map<String, String>
        */
   public static Map<String, String> maxProfit(int[] prices) {
       int maxGain = 0;
             int maxValue = -1;
        int i= prices.length -1  ;
       String sellingDays="";
       String buyingDays="";

       Map<String, String> res = new HashMap<>();

       while((i>=0)){
             if(prices[i] >= 0 && prices[i] >= maxValue) {
             //If the best selling day price occurs more than once in the
table, then all the days that contains the best selling prices are
concatenated to sellingDays string
                   sellingDays = (maxValue == prices[i] ?  sellingDays+" or "
+ String.valueOf(i+1) : String.valueOf(i+1));
                   maxValue = prices[i];
             }

             if(prices[i] >= 0 && (maxValue - prices[i]) >= maxGain){
                   //If the best buying day price occurs more than once in the
table, then all the days that contains the best selling prices are
concatenated to sellingDays string
                   buyingDays = (((maxValue - prices[i]) == maxGain)  ?
buyingDays+" or " + String.valueOf(i+1) :String.valueOf(i+1));
                       maxGain = maxValue - prices[i];
                   res.put("Buying Day", buyingDays);
                   res.put("Selling Day", sellingDays);
                   res.put("Maximum Gain", String.valueOf(maxGain));
                   }

             i--;
        }
        return res;
   }
   public static void main(String[] args) {
             int[] stock = {3,1,8,8,2,1,7,55,55,34};
             System.out.println(maxProfit(stock));
   }
}
```

}