The thinking is simple and is inspired by the best solution from Single Number II (I read through the discussion after I use DP).  
Assume we only have 0 money at first;  
4 Variables to maintain some interested 'ceilings' so far:  
The maximum of if we've just buy 1st stock, if we've just sold 1nd stock, if we've just buy 2nd stock, if we've just sold 2nd stock.  
Very simple code too and work well. I have to say the logic is simple than those in Single Number II.

**public** class Solution {

**public** **int** maxProfit(**int**[] prices) {

**int** hold1 = Integer.MIN\_VALUE, hold2 = Integer.MIN\_VALUE;

**int** release1 = 0, release2 = 0;

for(**int** i:prices){ // Assume we only have 0 money at first

release2 = Math.max(release2, hold2+i); // The maximum if we've just sold 2nd stock so far.

hold2 = Math.max(hold2, release1-i); // The maximum if we've just buy 2nd stock so far.

release1 = Math.max(release1, hold1+i); // The maximum if we've just sold 1nd stock so far.

hold1 = Math.max(hold1, -i); // The maximum if we've just buy 1st stock so far.

}

return release2; ///Since release1 is initiated as 0, so release2 will always higher than release1.

}

}