PRACTICAL 8

AIM:-WAP and Algorithm of knapsack

//ALGORITHM

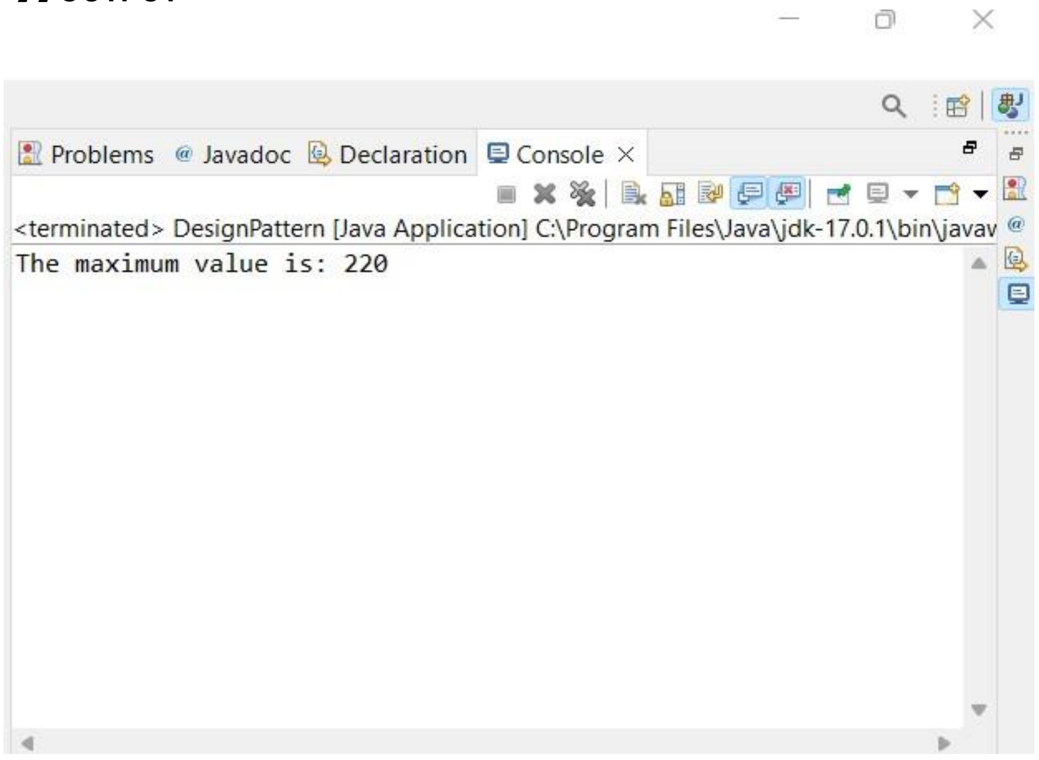
Fractional Knapsack (Array v, Array w, int W)

```
1. for i= 1 to size (v)
2. do p [i] = v [i] / w [i]
3. Sort-Descending (p)
4. i ← 1
5. while (W>0)
6. do amount = min (W, w [i])
7. solution [i] = amount
8. W= W-amount
9. i ← i+1
10. return solution
```

//PROGRAM

```
#include <bits/stdc++.h>
using namespace std;


int max(int a, int b) { return (a > b) ? a : b; }


int knapSack(int W, int wt[], int val[], int n)
{

    if (n == 0 || W == 0)
        return 0;

    if (wt[n - 1] > W)
        return knapSack(W, wt, val, n - 1);

    else
        return max(
                val[n - 1]
                    + knapSack(W - wt[n - 1],
                               wt, val, n - 1),
                knapSack(W, wt, val, n - 1));
    }

    int main()
    {
        int val[] = { 60, 100, 120 };
        int wt[] = { 10, 20, 30 };
        int W = 50;
        int n = sizeof(val) / sizeof(val[0]);
        cout << knapSack(W, wt, val, n);
        return 0;
}
```

//OUTPUT



```
<terminated> DesignPattern [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javav
The maximum value is: 220
```

//COMPLEXITY

Time Complexity: O(N*W).