Name – Rahul Bhor

Roll No – 05
Batch – T1

- -

Problem Statement - Write a program to solve a 0-1 Knapsack problem using dynamic programming or branch and bound strategy.

- -

Knapsack.java

```java
public class Knapsack {

    static int knapSack(int W, int[] wt, int[] val, int n) {
        int[] dp = new int[W + 1];

        for (int i = 1; i <= n; i++) {
            // Traverse weights backwards
            for (int w = W; w > 0; w--) {
                if (wt[i - 1] <= w) {
                    dp[w] = Math.max(dp[w], dp[w - wt[i - 1]] + val[i - 1]);
                }
            }
        }
        return dp[W];
    }

    public static void main(String[] args) {
        int[] val = {60, 100, 120};
        int[] wt = {10, 20, 30};
        int W = 50;
        int n = val.length;

        System.out.println(knapSack(W, wt, val, n));
    }
}
```

Output -

```
220
```