ASSIGNMENT NO. : 4

public class Knapsack {

public static int knapsack(int capacity, int[] weights, int[] values, int n) {

// Create a 2D array to store the maximum value for each subproblem

int[][] dp = new int[n + 1][capacity + 1];

// Build the dp array in bottom-up manner

for (int i = 1; i <= n; i++) {

for (int w = 0; w <= capacity; w++) {

// If the current item's weight is less than or equal to the current capacity

if (weights[i - 1] <= w) {

// Take the maximum of either taking the item or not

dp[i][w] = Math.max(dp[i - 1][w], dp[i - 1][w - weights[i - 1]] + values[i - 1]);

} else {

// If the item cannot be included, carry forward the previous value

dp[i][w] = dp[i - 1][w];

}}}

return dp[n][capacity]; // Maximum value for the given capacity }

public static void main(String[] args) {

// Sample items (values and weights)

int[] values = {60, 100, 120}; // Values of items

int[] weights = {10, 20, 30}; // Weights of items

int capacity = 50; // Maximum capacity of the knapsack

int n = values.length; // Number of items

// Calculate the maximum value that can be obtained

int maxValue = knapsack(capacity, weights, values, n);

System.out.println("Maximum value in the knapsack = " + maxValue);}}

**OUTPUT :**

Maximum value in the knapsack = 240