```java
public class KnapsackDP {
    // Function to solve the 0-1 Knapsack problem
    public static int knapsack(int[] weights, int[] values, int capacity) {
        int n = weights.length;
        int[][] dp = new int[n + 1][capacity + 1];

        // Building the DP table
        for (int i = 0; i <= n; i++) {
            for (int w = 0; w <= capacity; w++) {
                if (i == 0 || w == 0) {
                    dp[i][w] = 0; // Base case: no items or zero capacity
                } else if (weights[i - 1] <= w) {
                    // Include the item or exclude it, choose the maximum value
                    dp[i][w] = Math.max(values[i - 1] + dp[i - 1][w - weights[i - 1]], dp[i - 1][w]);
                } else {
                    dp[i][w] = dp[i - 1][w]; // Cannot include the item
                }
            }
        }

        // The bottom-right corner contains the maximum profit
        return dp[n][capacity];
    }

    // Main method
    public static void main(String[] args) {
        int[] weights = {10, 20, 30}; // Weights of the items
        int[] values = {60, 100, 120}; // Values of the items
        int capacity = 50; // Maximum capacity of the knapsack

        // Solve the knapsack problem
        int maxProfit = knapsack(weights, values, capacity);

        System.out.println("Maximum profit: " + maxProfit);

    }

}
```