Task 1:
 Knapsack Problem
Write a function int Knapsack(int W, int[] weights, int[] values) in Java that determines the maximum value of items that can fit into a knapsack with a capacity W. The function should handle up to 100 items. Find the optimal way to fill the knapsack with the given items to achieve the maximum total value. You must consider that you cannot break items, but have to include them whole.
ANS:

```
package com.Day20;
public class KnapsackProblem {

  // Function to find the maximum value of items that can fit into the
knapsack
  public static int knapsack(int W, int[] weights, int[] values) {
     int n = weights.length;
     int[][] dp = new int[n + 1][W + 1];
     // Build dp table
     for (int i = 1; i <= n; i++) {
        for (int w = 1; w <= W; w++) {
           if (weights[i - 1] <= w) {
              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];
           }
        }
     }
     // Find the maximum value
     return dp[n][W];
  }
  public static void main(String[] args) {
     int W = 10; // Knapsack capacity
     int[] weights = {2, 3, 4, 5}; // Weights of items
     int[] values = {3, 4, 5, 6}; // Values of items
     int maxValue = knapsack(W, weights, values);
     System.out.println("Maximum value that can be achieved: " +
maxValue);
```

```
    }
}
```

OUTPUT:

```
Maximum value that can be achieved: 13
```