BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
MUNSHI NAGAR, ANDHERI (WEST), MUMBAI – 400 058.
(Autonomous College Affiliated to University of Mumbai)
MASTER OF COMPUTER APPLICATIONS

Class : F.Y.MCA   Semester : II   Academic Year : 2024-25

Course Name : Design and Analysis of Algorithm MC507

Subject Incharge : Prof.Nikhita Mangaonkar

UCID: 2024510001   BATCH: A   NAME: Atharva Vasant Angre

EXPERIMENT NO: 04

EXPERIMENT TITLE: To implement Knapsack Algorithm.

4.1 To Implement Knapsack Algorithm (0/1 Knapsack).

Objective:

1. To Implement Knapsack Algorithm (0/1 Knapsack).

BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
MUNSHI NAGAR, ANDHERI (WEST), MUMBAI – 400 058.
(Autonomous College Affiliated to University of Mumbai)
MASTER OF COMPUTER APPLICATIONS

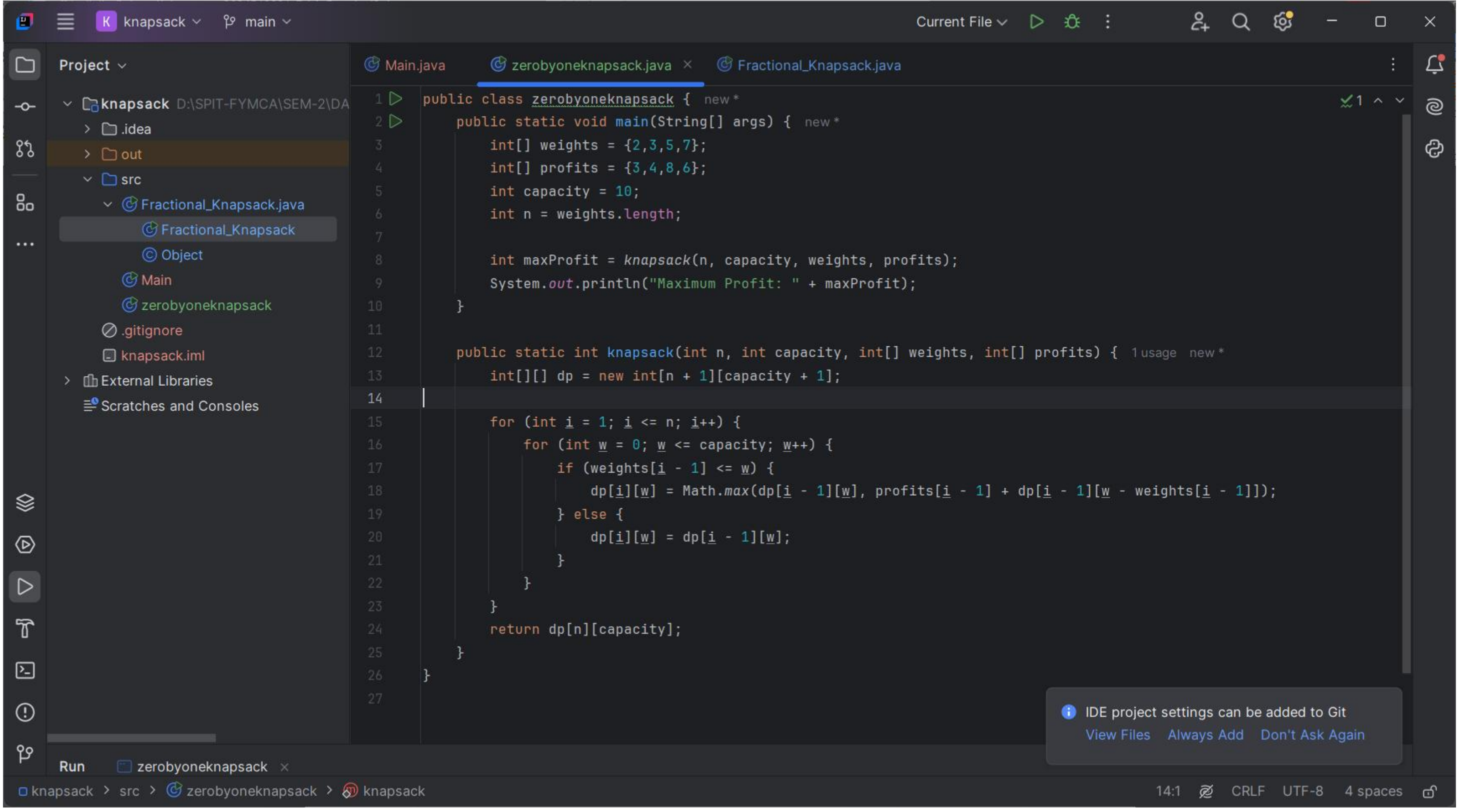
Class : F.Y.MCA Semester : II Academic Year : 2024-25

Course Name : Design and Analysis of Algorithm MC507

Subject Incharge : Prof.Nikhita Mangaonkar

Program code: -

0/1 Knapsack



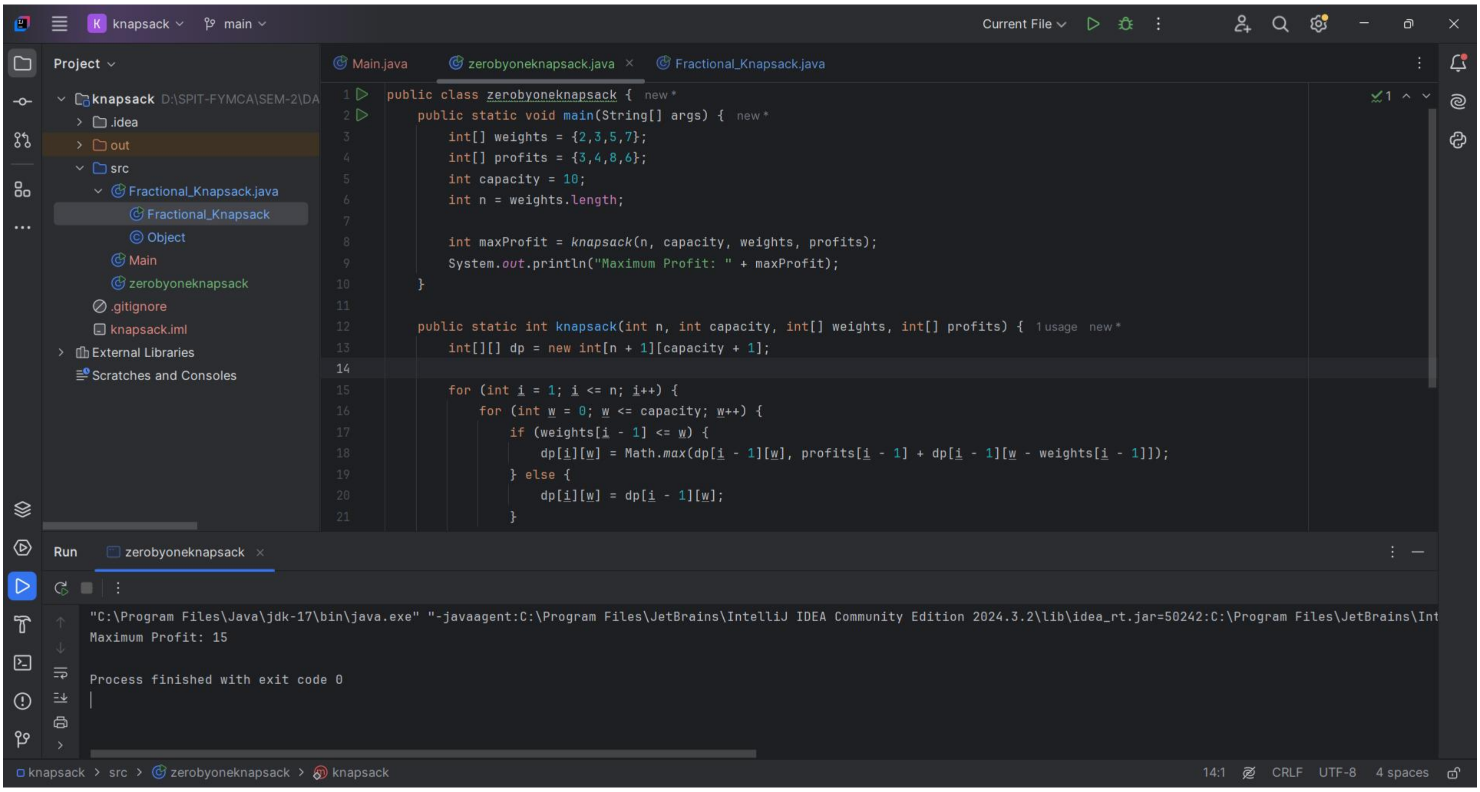
```java
public class zerobyoneknapsack {
    public static void main(String[] args) {
        int[] weights = {2,3,5,7};
        int[] profits = {3,4,8,6};
        int capacity = 10;
        int n = weights.length;

        int maxProfit = knapsack(n, capacity, weights, profits);
        System.out.println("Maximum Profit: " + maxProfit);
    }

    public static int knapsack(int n, int capacity, int[] weights, int[] profits) {
        int[][] dp = new int[n + 1][capacity + 1];

        for (int i = 1; i <= n; i++) {
            for (int w = 0; w <= capacity; w++) {
                if (weights[i - 1] <= w) {
                    dp[i][w] = Math.max(dp[i - 1][w], profits[i - 1] + dp[i - 1][w - weights[i - 1]]);
                } else {
                    dp[i][w] = dp[i - 1][w];
                }
            }
        }
        return dp[n][capacity];
    }
}
```

Class : F.Y.MCA Semester : II Academic Year : 2024-25

Course Name : Design and Analysis of Algorithm MC507

Subject Incharge : Prof.Nikhita Mangaonkar

Output:



```java
public class zerobyoneknapsack {  new *
    public static void main(String[] args) {  new *
        int[] weights = {2,3,5,7};
        int[] profits = {3,4,8,6};
        int capacity = 10;
        int n = weights.length;

        int maxProfit = knapsack(n, capacity, weights, profits);
        System.out.println("Maximum Profit: " + maxProfit);
    }

    public static int knapsack(int n, int capacity, int[] weights, int[] profits) {  1 usage  new *
        int[][] dp = new int[n + 1][capacity + 1];

        for (int i = 1; i <= n; i++) {
            for (int w = 0; w <= capacity; w++) {
                if (weights[i - 1] <= w) {
                    dp[i][w] = Math.max(dp[i - 1][w], profits[i - 1] + dp[i - 1][w - weights[i - 1]]);
                } else {
                    dp[i][w] = dp[i - 1][w];
                }
```

```
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.2\lib\idea_rt.jar=50242:C:\Program Files\JetBrains\Int
Maximum Profit: 15

Process finished with exit code 0
```

Conclusion:

The 0/1 Knapsack Algorithm helps in selecting items to maximize profit while keeping the total weight within the given capacity. It uses Dynamic Programming to ensure the best selection by considering each item’s weight and profit. This method guarantees an optimal solution but requires extra space. Overall, it is an efficient and widely used approach for solving resource allocation problems.