Name: Abhishek V Angadi

Roll No.: 14

Batch: B1

Class: A9

Practical No.3

Code:

```
import java.util.*;
public class temp {

   static void knapsack(float[] p, float[] w, float W){
      float[] pwratio = new float[p.length];
      for(int i=0;i<pwratio.length;i++){
         if(w[i]==0){
            pwratio[i]=Float.POSITIVE_INFINITY;
         }
         else{
            pwratio[i]=p[i]/w[i];
         }
      }
      float sort=0;

      for(int i=0;i<pwratio.length;i++){
         for(int j=0;j<pwratio.length-i-1;j++){
            if(pwratio[j]<pwratio[j+1]){
               sort=pwratio[j];
               pwratio[j]=pwratio[j+1];
               pwratio[j+1]=sort;

               sort=p[j];
               p[j] = p[j+1];
               p[j+1]=sort;

               sort=w[j];
```

```
            w[j]=w[j+1];
            w[j+1]=sort;
        }
    }
}

//////////////////// //sorting for min weight/////////////////////////////////////////////
    float[] minweight = Arrays.copyOf(w, w.length);
    float[] minweight_profit = Arrays.copyOf(p, p.length);

    float temp=0;
    for(int i=0;i<minweight.length;i++){
        for(int j=0;j<minweight.length-i-1;j++){
            if(minweight[j]>minweight[j+1]){
                temp=minweight[j];
                minweight[j]=minweight[j+1];
                minweight[j+1] = temp;

                temp=minweight_profit[j];
                minweight_profit[j]=minweight_profit[j+1];
                minweight_profit[j+1]=temp;
            }
        }
    }

////////////////////////////////////////////////////////////sorting max
profit//////////////////////////////
```

```
        float[] maxprofit = Arrays.copyOf(p, p.length);
        float[] maxprofit_weight=Arrays.copyOf(w, w.length);

        for(int i=0;i<pwratio.length;i++){
           for(int j=0;j<pwratio.length-i-1;j++){
              if(maxprofit[j]<maxprofit[j+1]){
                 sort=maxprofit[j];
                 maxprofit[j]=maxprofit[j+1];
                 maxprofit[j+1]=sort;

                 sort=maxprofit_weight[j];
                 maxprofit_weight[j] = maxprofit_weight[j+1];
                 maxprofit_weight[j+1]=sort;


              }
           }
        }
```

////////////////////////////////knapsack based on maxprofit////////////////////////////////

```
        float total_maxprofit=0;
        float total_maxprofit_weight=0;

        for(int i=0;i<maxprofit.length;i++){
```

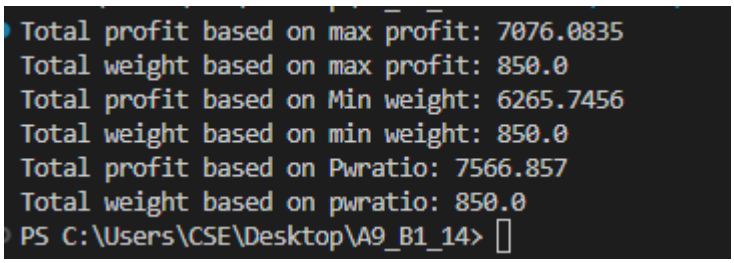
```
        if((total_maxprofit_weight +maxprofit_weight[i])<=W){
            total_maxprofit+=maxprofit[i];
            total_maxprofit_weight+=maxprofit_weight[i];
        }
        else{
            total_maxprofit+= (((W-
total_maxprofit_weight)/maxprofit_weight[i])*maxprofit[i]);
            total_maxprofit_weight+=W-total_maxprofit_weight;
            break;
        }
    }
    System.out.println("Total profit based on max profit: "+total_maxprofit);
    System.out.println("Total weight based on max profit: "+total_maxprofit_weight);



//////////////////////////////////////////based on min
weight//////////////////////////////////////////////////////////////////////

    float total_minweight_profit=0;
    float total_minweight_weight=0;

    for(int i=0;i<minweight.length;i++){
        if((total_minweight_weight+minweight[i])<=W){
            total_minweight_profit+=minweight_profit[i];
            total_minweight_weight+=minweight[i];
        }
        else{
```

```
            total_minweight_profit+= (((W-
total_minweight_weight)/minweight[i])*minweight_profit[i]);
            total_minweight_weight+=W-total_minweight_weight;
            break;
        }
    }
    System.out.println("Total profit based on Min weight: "+total_minweight_profit);
    System.out.println("Total weight based on min weight: "+total_minweight_weight);

////////////////////////////////////based on max
pwratio//////////////////////////////////////////////////////////////////////////////
    float total_pwratio_profit=0;
    float total_pwratio_weight=0;

    for(int i=0;i<pwratio.length;i++){
      if((total_pwratio_weight+w[i])<=W){
          total_pwratio_profit+=p[i];
          total_pwratio_weight+=w[i];
      }
      else{
          total_pwratio_profit+=(((W-total_pwratio_weight)/w[i])*p[i]);
          total_pwratio_weight+=W-total_pwratio_weight;
          break;
      }
    }

    System.out.println("Total profit based on Pwratio: "+total_pwratio_profit);
    System.out.println("Total weight based on pwratio: "+total_pwratio_weight);
```

```
////////////////////////////////////////////////////////////////////////////////////////////////
////////
```

```
    }
    public static void main(String args[]){
        float[] profit={360, 83, 59, 130, 431, 67, 230, 52, 93, 125, 670, 892, 600, 38, 48, 147,
78, 256, 63, 17, 120,
164, 432, 35, 92, 110, 22, 42, 50, 323, 514, 28, 87, 73, 78, 15, 26, 78, 210, 36, 85, 189, 274,
43, 33, 10, 19, 389, 276, 312};
        float[] weight={7, 0, 30, 22, 80, 94, 11, 81, 70, 64, 59, 18, 0, 36, 3, 8, 15, 42, 9, 0, 42,
47, 52, 32, 26, 48, 55,
6, 29, 84, 2, 4, 18, 56, 7, 29, 93, 44, 71, 3, 86, 66, 31, 65, 0, 79, 20, 65, 52, 13};

        knapsack(profit, weight, 850);
    }
}
```

OUTPUT:



Conclusion: Hence we successfully performed Knapsack for the given scenario and analysed which method gives the Most profit.

Github: https://github.com/Shadow3456rh