```
import java.util.Arrays;
import java.util.Scanner;

class Item implements Comparable<Item> {
    int value, weight;
    double ratio;

    public Item(int value, int weight) {
        this.value = value;
        this.weight = weight;
        this.ratio = (double) value / weight;
    }

    @Override
    public int compareTo(Item other) {
        return Double.compare(other.ratio, this.ratio);
    }
}

public class FractionalKnapsack {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter the number of items:");
        int n = scanner.nextInt();

        int[] values = new int[n];
        int[] weights = new int[n];

        System.out.println("Enter the values of the items:");
        for (int i = 0; i < n; i++) {
            values[i] = scanner.nextInt();
        }

        System.out.println("Enter the weights of the items:");
        for (int i = 0; i < n; i++) {
            weights[i] = scanner.nextInt();
        }

        System.out.println("Enter the maximum weight capacity of the
        knapsack:");
        int W = scanner.nextInt();

        System.out.println("Choose an option:");
        System.out.println("1. Solve Fractional Knapsack Problem");
        System.out.println("2. Exit");
        int choice = scanner.nextInt();

        if (choice == 1) {
            double maxValue = fractionalKnapsack(values, weights, W);
            System.out.println("Maximum value in the knapsack: " + maxValue);
        }
    }

    public static double fractionalKnapsack(int[] values, int[] weights, int W)
    {
        int n = values.length;
        Item[] items = new Item[n];

        for (int i = 0; i < n; i++) {
            items[i] = new Item(values[i], weights[i]);
        }

        Arrays.sort(items);

        double totalValue = 0;
        int currentWeight = 0;
```

```
        for (Item item : items) {
            if (currentWeight + item.weight <= W) {
                currentWeight += item.weight;
                totalValue += item.value;
            } else {
                int remainingWeight = W - currentWeight;
                totalValue += item.ratio * remainingWeight;
                break;
            }
        }

        return totalValue;
    }
}
```

```
Enter the number of items:
3
Enter the values of the items:
60 100 120
Enter the weights of the items:
10 20 30
Enter the maximum weight capacity of the knapsack:
50
Choose an option:
1. Solve Fractional Knapsack Problem
2. Exit
1
Maximum value in the knapsack: 240.0

Enter the number of items:
2
Enter the values of the items:
60 100
Enter the weights of the items:
10 20
Enter the maximum weight capacity of the knapsack:
50
Choose an option:
1. Solve Fractional Knapsack Problem
2. Exit
1
Maximum value in the knapsack: 160.0
```