```
import java.util.Arrays;
import java.util.Comparator;

public class FractionalKnapsack {
    // Inner class to represent an item with weight, value, and value-to-weight ratio
    static class Item {
        double weight, value, ratio;

        // Constructor to initialize an item with weight, value, and ratio
        public Item(double weight, double value) {
            this.weight = weight;
            this.value = value;
            this.ratio = value / weight;
        }
    }

    // Method to calculate the maximum value that can be carried in the knapsack
    public static double getMaxValue(Item[] items, double capacity) {
        // Sort items by their value-to-weight ratio in descending order
        Arrays.sort(items, new Comparator<Item>() {
            @Override
            public int compare(Item o1, Item o2) {
                return Double.compare(o2.ratio, o1.ratio);
            }
        });

        double totalValue = 0;

        // Iterate through the sorted items
        for (Item item : items) {
            if (capacity == 0) break;
```

```
        // If the item can be completely added to the knapsack
        if (capacity >= item.weight) {
            capacity -= item.weight;
            totalValue += item.value;
        } else {
            // If only part of the item can be added to the knapsack
            totalValue += item.value * (capacity / item.weight);
            capacity = 0;
        }
    }

    return totalValue;
}

// Main method
public static void main(String[] args) {
    // Array of items with specified weights and values
    Item[] items = {
        new Item(10, 60),
        new Item(20, 100),
        new Item(30, 120)
    };

    // Knapsack capacity
    double capacity = 50;

    // Calculate the maximum value that can be carried and print the result
    double maxValue = getMaxValue(items, capacity);
    System.out.println("Maximum value in knapsack: " + maxValue);
}
```

}