CSCI 232 Lab 12

Due Sunday April 28th @ 11:59 PM. Please submit this assignment (.java files) to the appropriate dropbox on D2L.

Background and Instructions

You will be implementing the **greedy algorithm** for the **fractional knapsack problem** (discussed during the greedy algorithms lecture). Given a list of items with their value and weight, and the capacity of the knapsack **N**, the goal is to fill the knapsack to maximize the total value of items taken without exceeding **N**. In the fractional version of the knapsack problem, you can take a "fraction" of an item to fill the remaining space of the knapsack

You will use Lab12Demo.java (linked below) as a starting point. You will need to define the the **`Item`** class and the necessary methods. Your code for the knapsack algorithm can go in the demo class. You don't need to add any other classes.

Here are the high-level steps of the knapsack algorithm with the greedy approach:

1. Sort the items from greatest to least, based on their value/weight ratio
2. Loop through the sorted array
    a. If the knapsack is full, exit the loop
    b. If it does not exceed the capacity of the knapsack, add it to the knapsack
    c. If the full item does exceed the capacitiy of the knapsack, but there is still room in the knapsack, then take a fraction of the item.

Your program will need to print out the items taken in the knapsack, along with the total cost and total weight of the knapsack. If a fraction of an item was taken, it should be indicated by **(F)** or something in the output.

Sample Output

When you run your program, it should look something like this:

```
Items placed in Knapsack
------------------------
-Name: Pearl | Value: 20.0 | Weight: 1.0
-Name: Emerald Bar | Value: 40.0 | Weight: 5.0
-Name: Silver Bar | Value: 88.0 | Weight: 12.0
-Name: Sapphire Bar | Value: 52.0 | Weight: 8.0
-Name: Gold Bar(F) | Value: 45.0 | Weight: 9.0
Total weight: 35.0
Total profit: 245.0
```

Starting Code

- **`Lab12Demo`:** **https://www.cs.montana.edu/pearsall/classes/spring2024/232/labs/Lab12Demo.java**

Hints

When you have a list of item that you are trying to sort, Java doesn't know what it means to sort `Item` objects. To have the Collections `.sort()` method sort by item ratio, you will need to implement the `Comparable` interface and override the compareTo() method:

Grading

- Item class is correctly defined – 3 points
- Your program places the correct items in the knapsack – 7 points

NOTE: If your code does not compile, correctness cannot be verified, and you won't receive any points for your code. Turn in code that compiles!