**Critical Thinking Assignment 2**

Jeremy Carney

Colorado State University Global

CSC400: Data Structures and Algorithms

Angelo Luo

May 25th, 2025

**Introduction**

This assignment extends the generic Bag class, a multiset supporting duplicate elements, by incorporating methods to compute the total element count, combine two bags, and extract unique elements. Implemented in Java, the enhanced Bag leverages a HashMap for efficient storage and retrieval. Testing utilizes Dungeons & Dragons-themed items—Fighter’s Honed Broadsword, Cleric’s Blessed Amulet, Mage’s Arcane Orb, and Thief’s Shadow Key—to demonstrate the new capabilities.

**Methodology**

The Bag class, built with a generic type <T>, uses a final HashMap<T, Integer> to store items and their counts. New methods were added: size() sums the counts of all items; merge(Bag<T> otherBag) integrates another bag’s elements by adding their counts; and distinct() creates a new bag with each item appearing once. Existing methods (add, remove, contains, count, toString) remain unchanged. The pseudocode below outlines the updated logic for clarity.

**BAG DATA STRUCTURE PSEUDOCODE**

• Set up the Bag

- Make a HashMap called 'items' to keep track of stuff and how many times it shows up.

- Mark 'items' as final so it can’t be swapped out.

• Add something

- Look if the thing is already in items.

- If it’s there, add 1 to its count.

- If not, give it a count of 1.

• Take something out

- Check if the thing is in items.

- If it is:

- If there’s only 1, get rid of it.

- If there’s more, subtract 1 from its count.

• See if something’s there

- Say true if the thing is in items, false if it’s not.

• Count how many of something

- If the thing’s in items, say its count.

- If not, say 0.

• Get the total size

- Add up the counts of everything in items.

- Give back the total.

• Merge another bag

- For each thing in the other bag:

- Add its count to the same thing in this bag.

- If it’s not in this bag, put it in with its count.

• Get distinct items

- Make a new bag.

- For each thing in items:

- Add it to the new bag with a count of 1.

- Give back the new bag.

• Show what’s in the bag

- Start a StringBuilder with "Bag{".

- For each thing in items:

- Add the thing, a colon, its count, and a comma.

- If there’s stuff in items, chop off the last comma.

- Finish with "}".

- Give back the StringBuilder as a string.

• Show it working

- Make two new bags, bag1 and bag2.

- Toss into bag1: "Fighter’s Honed Broadsword" (twice), "Cleric’s Blessed Amulet", "Mage’s Arcane Orb".

- Toss into bag2: "Cleric’s Blessed Amulet" (twice), "Thief’s Shadow Key", "Fighter’s Honed Broadsword".

- Print how many things in bag1 and bag2.

- Merge bag2 into bag1.

- Print bag1 after merging.

- Make a new bag with only distinct items from bag1.

- Print the distinct bag.

The test program instantiates two Bag<String> objects, populates them with items (including duplicates), displays their sizes, merges the second bag into the first, and generates a bag of distinct elements. Items include Fighter’s Honed Broadsword, Cleric’s Blessed Amulet, Mage’s Arcane Orb, and Thief’s Shadow Key.

*--Screenshots Below--*

A screen shot of a computer program

AI-generated content may be incorrect.

Figure : Java implementation of the extended Bag class - Part 1.

A screen shot of a computer screen

AI-generated content may be incorrect.

Figure : Java implementation of the extended Bag class - Part 2.

A screen shot of a computer program

AI-generated content may be incorrect.

Figure : Java implementation of the extended Bag class - Part 3.

**Results**

The enhanced Bag class functioned correctly. The first bag initially contained Fighter’s Honed Broadsword (2), Cleric’s Blessed Amulet (1), and Mage’s Arcane Orb (1), with a size of 4. The second bag held Cleric’s Blessed Amulet (2), Thief’s Shadow Key (1), and Fighter’s Honed Broadsword (1), with a size of 4. Merging the second bag into the first produced a bag with Fighter’s Honed Broadsword (3), Cleric’s Blessed Amulet (3), Mage’s Arcane Orb (1), and Thief’s Shadow Key (1). The distinct method created a new bag with each item once: Fighter’s Honed Broadsword (1), Cleric’s Blessed Amulet (1), Mage’s Arcane Orb (1), and Thief’s Shadow Key (1). The toString method accurately formatted outputs as Bag{item:count, ...}.

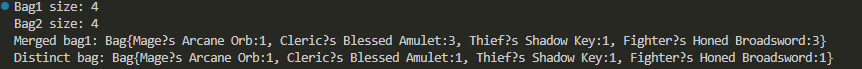


Figure : Console output demonstrating Bag operations and results.

**References**

GITHUB: <https://github.com/FistanRaist/CSC400-Data-Structures-and-Algorithms/tree/main/CTA2%20-%20Extended%20Java%20Bag>