Programing Project 2

Group 4:

(GitHub Link)

**1*.) If you used different arrays, structures, or classes to represent the different types of sets, would it be possible to have overloaded methods or operations that would provide the correct functionality regardless of whether or not you were using sets or multisets? Why or why not?***

Yes, overloading methods can work for sets and multisets, but it requires careful thought. In the ICallBullSet class, for example, we treat set operations like union, intersection, and difference for both sets and multisets. The challenge is that multisets track element frequencies, so methods like sum are specific to multisets. When overloading, you have to clearly define the differences between operations, especially when frequency comes into play. For example, a union for sets just adds unique elements, but a union for multisets has to consider how many times an element appears. So while overloading is possible, the logic for these methods needs to be adjusted based on whether you’re dealing with a set or a multiset.

***2.) Would it be possible to use the same data structure or class to store sets and multisets? Why or why not?***

Yes, it’s definitely possible to use the same data structure for both sets and multisets. A good example of this is the ICallBullSet class, which extends a HashMap. For sets, the keys in the map represent the elements, and the values are ignored (since sets don’t care about frequencies). For multisets, the values in the map represent the frequency of each element. However, you’d need to design the operations carefully depending on whether you're treating the collection as a set or a multiset. For instance, a union operation might just combine unique elements for sets but take frequency into account for multisets. So while it’s possible to use the same data structure, the behavior needs to adjust based on what type you're working with.

***3.) How easy or difficult is it to determine the type of set that you need to use based on the user's query? Why?***

Determining the right type of set based on a user’s input can be challenging because the intended operation is not always clear. In the FuzzyLogic class, for example, using negation would only make sense for fuzzy values, while union and intersection apply to both fuzzy values and traditional sets. Similarly, in the ICallBullSet class, operations like intersection work for sets, but sum is specific to multisets. Without additional context, distinguishing between these types may not always be straightforward.

***4.) Is it possible to store the data from one type of set, plain sets or multisets, in another type? Would you need to lose data in order to do so? Why?***

Yes, it's possible to store data from one type of set in another, but converting between the two types often involves some loss of data. For example, converting from a multiset to a set would lose frequency information, as sets do not track element occurrences. In the ICallBullSet class, the methods like sum and symmetricDifference handle multisets with element frequencies, but if we convert this to a plain set, we'd only keep the unique keys without their associated counts. On the other hand, converting from a set to a multiset would not lose data, but each element would have a count of one. The FuzzyLogic class works differently because it deals with continuous values instead of element presence or frequency, so converting fuzzy values to a standard set would also result in data loss.

**5.) Discuss what implications your answers to questions 1-4 have for someone trying to code an interface which would allow users to type in natural language queries.**

When coding an interface that lets users perform set operations, it is important to make sure the correct operations are applied to the correct set type. In the ICallBullSet class, operations like union and intersection are used for normal sets, while sum is meant for multisets. If the interface does not properly distinguish between them, users could apply the wrong operation to the wrong type of set. The FuzzyLogic class presents a different challenge, as it uses continuous values rather than discrete elements, meaning that operations like negation or union function differently from traditional set operations. Making sure that operations match the correct data type is crucial to ensuring that the system works as expected.

***6.) Discuss what implications your answers to questions 1-5 have for someone trying to code and interface which would allow users to access arbitrary types of data using natural language.***

If an interface is designed to handle different types of data, it needs to account for the differences between standard sets, multisets, and fuzzy sets. In the ICallBullSet and FuzzyLogic classes, each type of data structure follows different rules, and applying the wrong operations can lead to incorrect results. The system must be able to differentiate between these types and guide users toward the appropriate methods. For example, applying a multiset operation like sum to a normal set or trying to use a fuzzy operation on a standard set would not make sense. Ensuring that each data type is handled correctly is essential to making the system work properly.

**Contributions:**

**Edwin**: Wrote some of the code for the classes to implement the sets and multisets and revised the class for fuzzy sets to be instantiated as objects and started the test client.

**Corbin**: Made a few functions in the IcallBullSet class and added some testing in the main method.

**Markus**:

**Jaxon**: Questions/Report

**Jesus**: Made some revisions and added some test methods into the test client