Inventory System using Multiset Design

Introduction to the setting

The context for this inventory design is related to the game, the game this inventory will be implemented into is Top down Traditional roguelike. the setting will be a military complex set in the far future. the player character is a automatic mech built to scale the complex.

my inventory will need multiple features such as:

- multiple items
- same name items
- Expandable
- retractable
- sorting
- search function

therefore, my design needs to store items with the same name with information such as item amount and needs to be expandable and retractable.

for this reason, I'll be using Sequence<string>

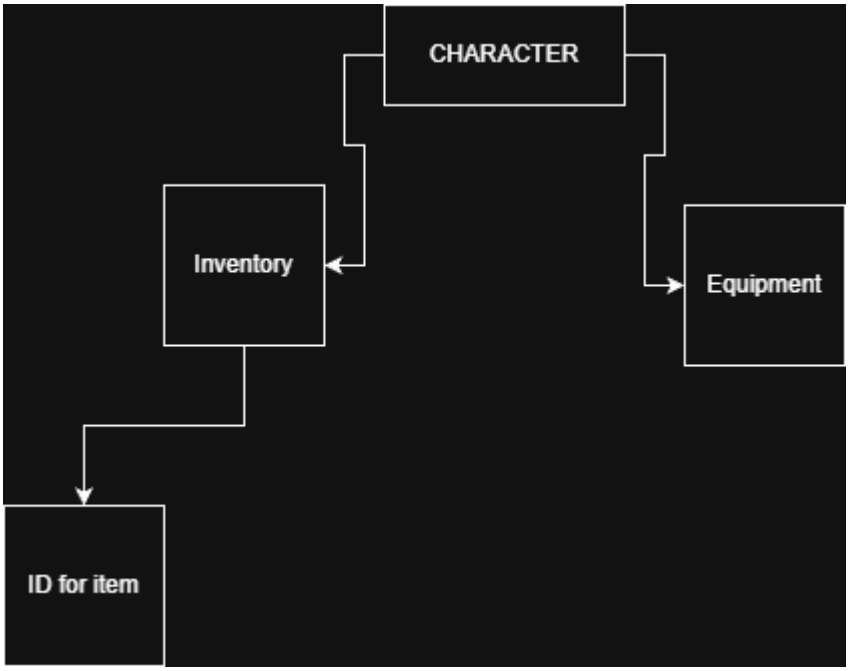
Why Sequence

The reason I am using Sequence is for an easier faster implementation within the project

Because my inventory needs to be expandable and retractable i feel that a Sequence set would be easist to modify the size of throughout the game.

using A sequence would lower the development time of my project by a signifigant amount

This Sequence will be used and interacted by the user, The inentory system will be connected to the player character as seen in this diagram



The ID system will allow for easy implementation and readibility for items and updates to items, as well as the implementation of new items

Operations

these are five operations that a multi set must have

Operation	Time Complexity
insert	O(1)
delete	O(N)
search	O(N)
clear	O(N)
get	O(N)

Insert: will be appended on to the end of the list if there is enough space in the inventory

delete: will search the list for an ID and remove item

Search: will search the list using a linear method to find all items containing searched text

clear: will clear all items from the inventory dropping them to the ground

get: will select the chosen item for use

Edge cases

when lowering inventory space any items that do not fit in the new space will be dropped

when deleting an item that occurs mutiple times it will prompt the user for the correct one same for get

the use of Sequence

the use of sequence allows for quick removal the top of the list which we be useful when changing inventory size

one constraint is searching will be slower than other methods

Set Operations

my games invintory will have a couple interactions

Operation
union
difference

Union

in my game union will be used to quick stack items from any source to the inventory and vice versa, this allows for quick inventory management with other systems

union will use get and delete to remove items from the sequence and move them to the other system

difference

difference will be used to show the difference in inventory this could be used to show what new items you could get from an enemy or what new items would be valuable to trade for in game

the implementation would be to iterate through the sequence and log all the items then iterate through the other list and print only items that dont appear

Extension feature

there will be a getReference() method that returns the item ID's related information such as stats rarity and cost

the game will feature a method called scrap() scrap will take one item and based on its rarity it will give you armor value here is a general since of how it would be implemented in psuedo code

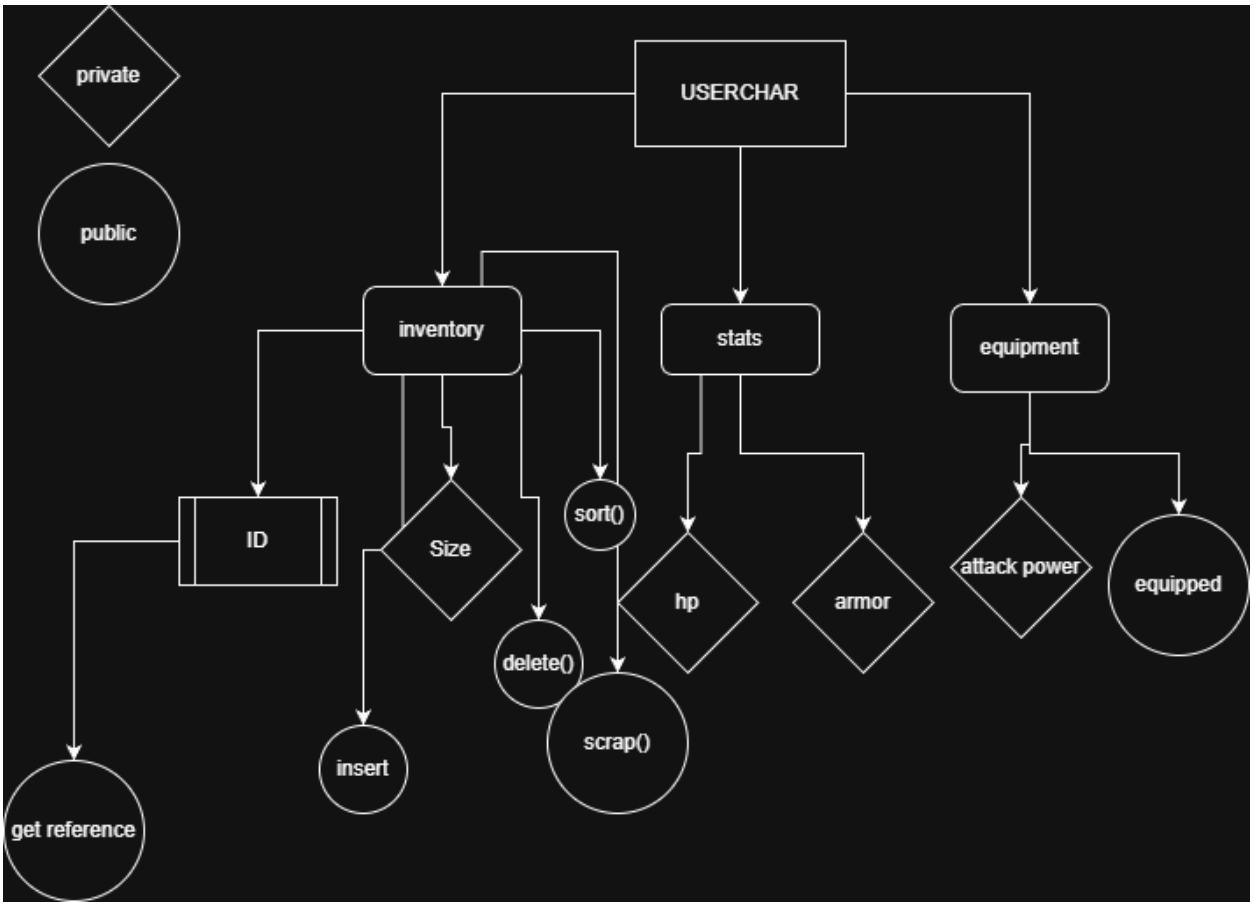
```
scrap(ID item){
if item.getReference()==common
        item.delete()
        return 10
else if item.getReference()==uncommon
        item.delete()
        return 20
else if item.getReference()==rare
        item.delete()
        return 30
else item.getReference()==legendary
        item.delete()
        return 50
}
```

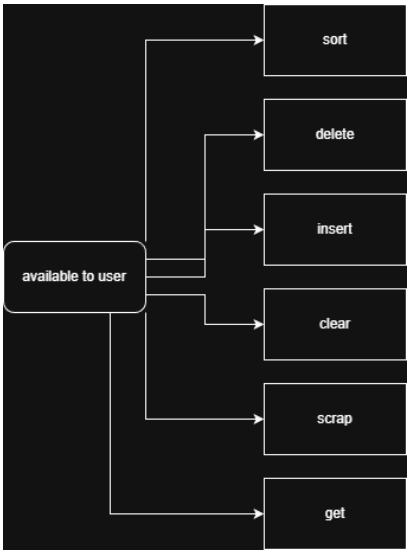
this would allow easy access to modifying armor this would run in the character class

this would also play into the theme of an ever changing inventory.

A small rundown of the code is that it gets and item and then returns the associated value to the level of rarity the item is, then deletes the item

UML





Trade off

I could have chosen to use a hashTable for my multiset which I feel would be more appropiate than a Binary Search Tree.

A hashtable would have had a faster search and delete function with the same space complexity (kurtis), but the rade off would be development time

A Hashtable would also not be simple to implement the sort function or other functions and methods.

> Undefined behavior is chaos. I don't want chaos in my software. The better I can predict how my code will behave, the less time I'll spend debugging it.
>
> – Jeffery Rennie

A Hashtable would also have the potential for a longer insert as my current implementation is to append it to the end it would also be harder to modify the size

implementation of a hash table

If i were to implement a hash table I would change a couple things

first I would remove the sort function, it would take to long to implement and Would not be worth the hastle to implement it

my hashtable would be a random index hashtable as I have made one before

I would probe the hash Table using the index when searching for an item or deleting an item

one major problem is the list would need new indexing often with how my inventory system is able to increase and decrease its max size

overall a hashtable would be a poor fit for the inventory I plan to design

Conclusion

in conclusion using a sequence for my inventrory system would be the best fit for my project.

this is due to the amount of size modifications i need for the list while still restrcting the size for the user

the inventory also fits the setting of an everchanging automaton well, which is my main goal for the system

Citations

Rennie, J. (n.d.). Why I avoid using hash tables. most of the programming languages I… | by Jeffrey Rennie | Medium. https://surferjeff.medium.com/why-i-avoid-using-hash-tables-3bf5734fafb6

Larson, P. (n.d.). Dynamic hash tables | communications of the ACM. Association for Computing Machinery, Digital Library. https://dl.acm.org/doi/abs/10.1145/42404.42410

Kartik. (2024, November 2) What does “space complexity” mean? GeekforGeeks. https://www.geeksforgeeks.org/dsa/g-fact-86/