Nolan Anderson

CPE 212-01 SP20

Josh Langford

03/06/2020

Project 04 Documentation

Character.cpp

-Method name: Character::AddToInventory(Item &i)

-Returns: Function type void, no return and only modifies.

-Pre condition: The class Item and Character has been initialized and constructed in order to be called into the function.

-Post condition: The object i of class type Item will be added to the character's items.

-Method name: Character::operator==(const Character &otherCharacter)

-Returns: Boolean value

-Pre condition: There are two characters that have been declared / created and can be compared.

-Post condition: The boolean value returned depends on Character1.uid == Character2.uid

-Method name: Character::operator != (const Character &otherCharacter)

-Returns: Boolean value

-Pre condition: There are two characters that have been declared / created and can be compared.

-Post condition: The boolean value returned depends on Character1.uid != Character2.uid

-Method name: Character::operator < (const Character &otherCharacter)

-Returns: Boolean value

-Pre condition: There are two characters that have been declared / created and can be compared.

-Post condition: The boolean value returned depends on Character1.uid < Character2.uid

-Method name: Character::operator > (const Character &otherCharacter)

-Returns: Boolean value

-Pre condition: There are two characters that have been declared / created and can be compared.

-Post condition: The boolean value returned depends on Character1.uid > Character2.uid

-Method name: Character::LootTarget(Character &target)

-Returns: Function type void, no return and only modifies.

-Pre condition: A target character has been declared with items (or none) and can be called into the function.

-Post condition: The items from the target object of type Character will be added to the players inventory.

Inventory.cpp

-Method name: Inventory::Add(const Item &i)

-Returns: Function type void, no return and only modifies.

-Pre condition: The Item and Inventory classes have been defined and constructed.

-Post condition: The object i of class type Item will be added to the inventory.

-Method name: Inventory::Remove(const Item &i)

-Returns: Function type void, no return and only modifies.

-Pre condition: The Item and Inventory classes have been defined and constructed and there is at least one item in the Inventory.

-Post condition: The object i of class type Item will be removed if found from the inventory.

-Method name: Inventory::isEmpty() const

-Returns: Boolean value

-Pre condition: There is an inventory class that has been declared and constructed.

-Post condition: The boolean truth value is determined by the state of the first object in the list. If it is NULL, then the function will return true.

-Method name: Inventory::isFull() const

-Returns: Boolean Value

-Pre condition: There is an inventory class that has been declared and constructed and is >0 length.

-Post condition: The boolean truth value is determined by the state of the last object in the list. If the position is equal to the max size of the list, then the function will return true. Else false.

-Method name: Inventory::ShowInventory() const

-Returns: Function type void, does not return values.

-Pre condition: There is an inventory class that has been declared and constructed and there are objects in the inventory.

-Post condition: For the items that are in the inventory, they will be outputted using the PrintItem() function.

-Method name: Inventory::PopFront() const

-Returns: Returns an item of type Item

-Pre condition: There is an inventory class that has been declared and constructed and there is at least one item at the front.

-Post condition: Deletes and returns the item that is at the front of the list.

Item.cpp

-Method name: Item::operator==(const Item &otherItem) const

-Returns: Boolean value

-Pre condition: There are two items that have been declared and can be compared.

-Post condition: The boolean value returned is determined by the outcome of comparing: item1 == item2

If item1 == item2: return true

else: return false

-Method name: Item::operator!=(const Item &otherItem) const

-Returns: Boolean value

-Pre condition: There are two items that have been declared and can be compared.

-Post condition: The boolean value returned is determined by the outcome of comparing: item1 != item2

If item1 != item2: return true

else: return false

-Method name: Item::operator>(const Item &otherItem) const

-Returns: Boolean value

-Pre condition: There are two items that have been declared and can be compared.

-Post condition: The boolean value returned is determined by the outcome of comparing: item1 > item2

If item1> item2: return true

else: return false

-Method name: Item::operator<(const Item &otherItem) const

-Returns: Boolean value

-Pre condition: There are two items that have been declared and can be compared.

-Post condition: The boolean value returned is determined by the outcome of comparing: item1 < item2

If item1 < item2: return true

else: return false

-Method name: Item::operator<=(const Item &otherItem) const

-Returns: Boolean value

-Pre condition: There are two items that have been declared and can be compared.

-Post condition: The boolean value returned is determined by the outcome of comparing: item1 <= item2

If item1 <= item2: return true

else: return false

-Method name: Item::operator>=(const Item &otherItem) const

-Returns: Boolean value

-Pre condition: There are two items that have been declared and can be compared.

-Post condition: The boolean value returned is determined by the outcome of comparing: item1 >= item2

If item1 >= item2: return true

else: return false

Party.cpp

-Method name: std::string Party::GetName() const

-Returns: Observer function, returns the party name.

-Pre condition: The Party class has been constructed.

-Post condition: The Party Name is simply returned.

-Method name: void Party::AddMember(Character *c)

-Returns: Function type void, does not return any value and only modifies.

-Pre condition: The character and party class have been declared so c can become a new object.

-Post condition: The object “c” of type Character will be added to the list of party members.

-Method name: void Party::RemoveMember(Character *c)

-Returns: Function type void, does not return any value and only modifies.

-Pre condition: The character and party class have been declared so that c can become a new object. There are > 0 members in the party

-Post condition: The object “c” of type Character will be removed from the list of party members if found.

-Method name: void Party::ShowParty(Character *c)

-Returns: Function type void, does not return any value and only outputs data.

-Pre condition: The character and party class have been declared so that we can declare the new object c and there needs to be characters in the party.

-Post condition: The function will output the data held in “c”.

-Method name: Character* Party::FindMember(Character *c)

-Returns: Function type Character, will return an object of type Character.

-Pre condition: The Party class and Character class have been declared so that c can be called in and there are >0 members in the party.

-Post condition: The object “c” of class type Character will be returned if found, NULL if not found.

-Method name: void Party::Attack(Party* targetGroup, int attackerUID, int targetUID)

-Returns: Function type void, does not return and only modifies the object that are called in.

-Pre condition: The party class has been declared and constructed and there are characters in the party.

-Post condition: The objects called in will be updated based on the data that is held within the other objects. I.e. The data in the target will take the “damage” of the data in the attacker.

-Method name: int Party::GetSize() const

-Returns: Returns an integer value of the size of the party

-Pre condition: The Party class has been declared and constructed.

-Post condition: The number returned depends on the size of the Party

List_impl.hpp

-Method name: List<Type>::List()

-Returns: Function type constructor, does not return.

-Pre condition: The class list is declared in the list.hpp file.

-Post condition: The class List will constructed

-Method name: List<Type>::~List()

-Returns: Function type deconstructor, does not return.

-Pre condition: The class is declared in the list.hpp file.

-Post condition: The class list will be deconstructed and all data removed.

-Method name: List<Type>::AddItemSorted(const Type &data)

-Returns: Function type void, only modifies the list.

-Pre condition: There is a list that “data” can be added into.

-Post condition: The item data will be added into the list in ascending order.

-Method name: List<Type>::DeleteItem(const Type &data)

-Returns: Returns a boolean value

-Pre condition: There are items in a list that can be removed and are found by using the “data” object that is called in.

-Post condition: If “data” is found and deleted, return true.

Else return false.

-Method name: List<Type>::Count() const

-Returns: unsigned integer value

-Pre condition: There is a list that has been declared with or without items in it

-Post condition: The unsigned integer value returned depends on the size of the list currently.

-Method name: List<Type>::Front()

-Returns: A reference to a varying data type.

-Pre condition: There is a list with an object at the front.

-Post condition: The reference to the data returned depends on the type of the data that is at the beginning of the list.

-Method name: List<Type>::Front() const

-Returns: A constant value to a varying data type.

-Pre condition: There is a list with an object at the front.

-Post condition: The value of the data returned depends on the type of the data that is at the beginning of the list.

-Method name: List<Type>::Back()

-Returns: A reference to a varying data type.

-Pre condition: There is a list with an object at the end.

-Post condition: The reference to the data returned depends on the type of the data that is at the end of the list.

-Method name: List<Type>::Back() const

-Returns: A constant value to a varying data type.

-Pre condition: There is a list with an object at the end.

-Post condition: The value of the data returned depends on the type of the data that is at the end of the list.

-Method name: List<Type>::IterateItems() const

-Returns: a pointer to iterator type.

-Pre condition: There is a list with length > 1 with items that the iterator can point to.

-Post condition: The iterator will be paired with the next object in the list

-Method name: List<Type>::AtEnd() const

-Returns: boolean value

-Pre condition: There is an iterator paired with an object in the list.

-Post condition: The boolean value returned depends on whether or not the iterator is at the end of the list.

-Method name: List<Type>::ResetIterator() const

-Returns: Function type void, does not return a type.

-Pre condition: There is an iterator value.

-Post condition: The iterator value will be returned to the beginning.

-Method name: List<Type>::PopFront()

-Returns: An object of a varying type that is at the front of the list

-Pre condition: There is a list declared with at least one object in it

-Post condition: The value at the front of the list will be removed and then returned to the caller/