**Spike:** 4

**Title:** Inventory

**Author:** Ryan Chessum, 102564760

**Goals / deliverables:**

* Code, see /09 – Spike - Game Data Structures/Task 9 Inventory/
* Short Report
* Spike Report

**Technologies, Tools, and Resources used:**

List of information needed by someone trying to reproduce this work

* Visual Studio 2019
* C plus plus reference (<https://www.cplusplus.com/reference/>)
* Zorkish game specifications

**Tasks undertaken:**

* Download and install Visual Studio
* Create a new C++ project
* Create identifiable object class
* Create game object class
* Create item class
* Create inventory
* Create player
* tests

**What we found out:**

To create the inventory system in Zorkish I created an inventory class which has a vector member variable that stores pointers to Item objects.

First I create the items to be stored in the inventory. I plan on having most classes used for the Zorkish game implementation being child classes of several base objects.

First there is the Identifiable object. This class has a vector of ids that we can use to identify what this object is using it’s member functions. It has member functions to see if the object has an identifier, get the first identifier and also to add identifiers. This will be used for a wide range of things including commands and objects in the game.

Text

Description automatically generated

Next is the game object which is a child class of the identifiable object class. It adds name and description variables as well as functions to get different sized descriptions. This class is for things like items, entities, the player, locations, etc.

A screenshot of a computer

Description automatically generated with medium confidence

Now we can add items. This class is pretty simple and doesn’t change much. This is to specify the types of game objects that we want in our inventory. If we let our inventory just store regular game objects then it would be able to store things like locations which we don’t want. This also lets us make items sub classes for things like weapons, bags, etc. which can add more functionality but since they inherit from the item class they would be able to be stored in the inventory.

Text

Description automatically generated

After I added the item class I could create the inventory class. It has a vector that points to items. It also has a few member functions that manage the inventory for us.

Text

Description automatically generated

First the has item function takes in an id and checks to see if it has an item that matches the id. This can be used for commands like looking at or taking items. We need to see if we have it first.

Text

Description automatically generated

Next is the put function which adds an item pointer to the vector.

Text

Description automatically generated

The fetch function returns a pointer to an item in our inventory based on the id passed in. This could be used when a player wants to use an item or look at an item.

Text

Description automatically generated

The take function fetches an item and returns its value (not a pointer) while also removing it from the inventory. This could be used when a player uses a one use item or transfers an item between inventories like taking an item out of a location or putting it somewhere in one.

Text

Description automatically generated

Lastly the item list function gets every item stored in the inventory and returns a list of decryptions as a string. This can be used when a player looks at their inventory.

Text

Description automatically generated

Now we can add a player class and give it an inventory object.

Text

Description automatically generated

I made a few tests to see if everything worked as intended and it did.

Text

Description automatically generated

Text

Description automatically generated

Graphical user interface, application, Teams

Description automatically generated

Graphical user interface, application, email

Description automatically generated