Spike: 09
Title: Composite Pattern

Author: Ford Killeen, 9731822

Goals / deliverables:
Goals this spike aims to achieve:

- Extend upon Zorkish by adding a composite pattern so that locations can contain entities
- Load adventure files with game entities
- Players can modify entities using look, take, put and open commands

Deliverables required:

- Code showcasing the new entities and their use
- Spike report

Technologies, Tools, and Resources used:
The following is required to complete this spike:

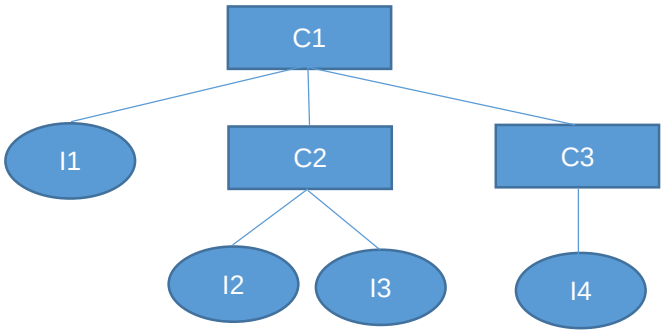
- Visual Studio 2015
- Zorkish game specification
- Online C++ references
- A text file adventure of your own making with added entities

Tasks undertaken:
The list below details the steps taken to complete this spike.

- Firstly I designed the classes that would build the core of the composite pattern. That is the base `Component` class, which is inherited by the `Item` and `Container` classes, similar to the Leaf and Branch classes as seen in other examples.

- After coding these main classes, I set up a test in `main.cpp` to check that it was working as intended.

```
Container c1 = Container();
c1.set_name("c1");
Container *c2 = new Container();
c2->set_name("c2");
Container *c3 = new Container();
c3->set_name("c3");
Item *i1 = new Item();
i1->set_name("i1");
Item *i2 = new Item();
i2->set_name("i2");
Item *i3 = new Item();
i3->set_name("i3");
Item *i4 = new Item();
i4->set_name("i4");
c1.add("i1", i1);
c2->add("i2", i2);
c2->add("i3", i3);
c3->add("i4", i4);
c1.add("c2", c2);
c1.add("c3", c3);
c1.action(Action::OPEN);
```




```
Container:c1 contains 3
Container:c2 contains 2
Item: i2
Item: i3
Container:c3 contains 1
Item: i4
Item: i1
```

- Once the test was run and looking positive, I added some new classes that would inherit from either the `Item` or `Container` classes. I also made the `Location` class inherit from the `Container` class so that it could contain entities and be able to add or remove entities.
- At this point I discovered a few small complications. The first being that I had to figure out a way to handle multiple actions within the one action() method, and how to determine which object should be actioned against. See the Issues section below for more detail.

What we found out:

By completing this spike I found out about the Composite Pattern, which seems like it would be a useful pattern for certain situations, but unfortunately I could not find this usefulness in the Zorkish game context.

Issues:

There were a few major issues when attempting to complete this spike. The first being that from what I could see online and understand in the composite pattern, it is supposed to be used where an item could either be a branch or a leaf in a tree, either containing more objects or being the end of a line. I also understood that the items should possess a function that can be called independent of whether the item contains more items or it is just the end of a branch. This made it hard as I has an `action`() method which could be called on any of my objects, but certain objects like a `Chest`, would have multiple actions that could be taken (open, look at, look in, unlock). From my understanding of this pattern, it would go against the pattern to add other functions, like an open method to the classes, as an Item that does not contain other objects cannot be opened. I therefore created an `Action` enum

and used that as a parameter to determine what action should be taken. This seemed to solve the problem I had regarding what action should be taken.

```
enum Action { OPEN, LOOKIN, LOOKAT, UNLOCK };
```

```
virtual void action(Action _action) = 0;
```

I then ran into another problem, which was determining which objects the action should be called on. From my understanding of the composite pattern, the action method would be called on one object and it would then call that same method on all of its contained objects. One way to possibly solve this could be to add the name of the object we care about as a parameter to the action method, but this seemed a bit overkill and incorrect. I then decided to leave this spike where I got to as I couldn't find a way to solve my problems while sticking closely to the composite pattern as I understood it. I will seek help on this in future labs and continue on with other spikes in the meantime.