KingSchlock / UploadedCodeToPrintAgain Private

Code Issues Pull requests Actions Projects Security Insights Settings

main

UploadedCodeToPrintAgain / IHaveInventory.cs / <> Jump to

KingSchlock Add files via upload

1 contributor

20 lines (18 sloc) 417 Bytes

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SwinAdventure
{
    public interface IHaveInventory
    {
        //! Everything that has an inventory has to have the ability to
        //! locate items.
        GameObject Locate(string id);

        public string Name
        {
            get;
        }
    }
}
```

KingSchlock / UploadedCodeToPrintAgain Private

Code Issues Pull requests Actions Projects Security Insights Settings

main

UploadedCodeToPrintAgain / Bag.cs / Jump to

KingSchlock Add files via upload

1 contributor

31 lines (26 sloc) | 727 Bytes

```csharp
namespace SwinAdventure
{
    public class Bag : Item, IHaveInventory
    {
        Inventory _inventory = new();

        public Bag(string[] idents, string name, string description) : base(idents, name, desc
        {

        }

        public Inventory Inventory
        {
            get { return _inventory; }
        }

        public override string FullDescription
        {
            get { return $"In the {Name} you can see\n{_inventory.ItemList}"; }
        }

        public GameObject Locate(string id)
        {
            if(this.AreYou(id) == true)
            {
                return this;
            }
            return _inventory.Fetch(id);
        }
    }

}
```

KingSchlock / UploadedCodeToPrintAgain Private

Code Issues Pull requests Actions Projects Security Insights Settings

main

UploadedCodeToPrintAgain / Player.cs / Jump to

KingSchlock Add files via upload

1 contributor

39 lines (35 sloc) | 1.31 KB

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SwinAdventure
{
     public class Player : GameObject, IHaveInventory //x TODO Implement Inventory field, prope
    {
        Inventory _inventory = new Inventory();
         public Player(string name, string description) : base(new string[] {"me", "inventory"}
        {


        }

        public Inventory Inventory
        {
            get { return _inventory; }
        }

        public override string FullDescription //! Can only override virtual properties
        {
             get { return $"You are {Name} {base.FullDescription}.\nYou are carrying\n{Inventor
        }
        public GameObject Locate(string id) //! Checks if the player holds an object with id
        {
            if (this.AreYou(id) == true)
            {
                return this; // returns this object

            }
             return _inventory.Fetch(id); // if the object isn't around then check our inventor
            /*! NOTE:
             *      The Locate operation should return null if no objects match id as the defa
             *      Fetch is null.
             */
        }
    }
}
```

KingSchlock / **UploadedCodeToPrintAgain** Private

Code Issues Pull requests Actions Projects Security Insights Settings

main

UploadedCodeToPrintAgain / **LookCommand.cs** / Jump to

KingSchlock Add files via upload

1 contributor

81 lines (68 sloc) | 2.16 KB

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SwinAdventure
{
    public class LookCommand : Command
    {
        public LookCommand() : base(new string[] { "look" })
        {

        }

        //! A series of checks which run when the look command is used
        //! Returns the same as LookAtIn
        public override string Execute(Player player, string[] text)
        {
            IHaveInventory container;
            string thingId;

            if (text.Length != 3 && text.Length != 5)
            {
                return "I don't know how to look for that.";
            }

            if (text[0] != "look")
            {
                return "Error in look input";
            }

            if (text[1] != "at")
            {
                return "What do you want to look at?";
            }

            if (text.Length == 5 && text[3] != "in")
            {
                return "What do you want to look in?";
            }


            if (text.Length == 3)
            {
                container = player;
            }
            else
            {
                container = FetchContainer(player, text[4]);
            }

            if (container == null)
            {
                return $"I can't find the {text[4]}";
            }

            thingId = text[2];
            return LookAtIn(thingId, container);
        }

        //! Grabs a container based on a string
        private IHaveInventory FetchContainer(Player player, string containerId)
        {
            return player.Locate(containerId) as IHaveInventory;
        }

        //! checks if the thing requested exists inside a container, if so return it's full de
        private string LookAtIn(string thingId, IHaveInventory container)
        {
            if(container.Locate(thingId) == null)
            {
                return $"I can't find the {thingId}";
            }
            else
            {
                return container.Locate(thingId).FullDescription;
            }
        }
    }
}
```

KingSchlock / **UploadedCodeToPrintAgain** Private

Code Issues Pull requests Actions Projects Security Insights Settings

main

UploadedCodeToPrintAgain / **TestLookCommand.cs** / <> Jump to

KingSchlock Add files via upload

1 contributor

116 lines (98 sloc) | 4.19 KB

```csharp
using SwinAdventure;

namespace SwinAdventureTests
{
    [TestFixture()]
    public class TestLookCommand
    {
        LookCommand lookTest;
        Player playerTest;
        Bag bagTest;
        Item swordTest;

        string unknown, noBag,
            badLength, badLook,
            badAt, badIn;

        [SetUp()]
        public void Setup()
        {
            lookTest = new();
            playerTest = new("thomas", "The mighty keyboard warrior");
             bagTest = new(new string[] { "satchel" }, "satchel", "it's smol");
            swordTest = new(new string[] { "sword" }, "sword", "lil poker");

            unknown = "I can't find the sword";
            noBag = $"I can't find the {bagTest.Name}";

            badLength = "I don't know how to look for that.";
            badLook = "Error in look input";
            badAt = "What do you want to look at?";
            badIn = "What do you want to look in?";

            playerTest.Inventory.Put(swordTest);
        }

        //! Return players descript when looking at the inventory
        [Test()]
        public void TestLookAtMe()
        {
            Assert.That(lookTest.Execute(playerTest, new string[] {"look", "at", "me"}),
                Is.EqualTo(playerTest.FullDescription));
        }

        //! Returns item description when looking for an item in players invent
        [Test()]
        public void TestLookAtItem()
        {
            Assert.That(lookTest.Execute(playerTest, new string[] {"look", "at", "sword"}),
                Is.EqualTo(swordTest.FullDescription));
        }

        //! Responds unknown when item isn't in inventory
        [Test()]
        public void TestLookAtUnkn()
        {
            playerTest.Inventory.Take("sword");

             Assert.That(lookTest.Execute(playerTest, new string[] { "look", "at", "sword", "in
                Is.EqualTo(unknown));
        }

        //! Returns item description when searching for item specifically in invent
        [Test()]
        public void TestLookAtItemInInventory()
        {
             Assert.That(lookTest.Execute(playerTest, new string[] {"look", "at", "sword", "in"
                Is.EqualTo(swordTest.FullDescription));
        }

        //! Returns item description when searching for it in a bag in players invent
        [Test()]
        public void TestLookAtItemInBag()
        {
            playerTest.Inventory.Take("sword");

            bagTest.Inventory.Put(swordTest);
            playerTest.Inventory.Put(bagTest);

             Assert.That(lookTest.Execute(playerTest, new string[] {"look","at","sword","in","s
                Is.EqualTo(swordTest.FullDescription));
        }

        //! Returns noBag when there's no container in players invent
        [Test()]
        public void TestLookAtItemInNoBag()
        {
            Assert.That(lookTest.Execute(playerTest, new string[] { "look", "at", "sword", "in
                Is.EqualTo(noBag));
        }

        //! Returns unknown when requested item isn't in bag
        [Test()]
        public void TestLookAtNoItemInBag()
        {
            playerTest.Inventory.Put(bagTest);

            Assert.That(lookTest.Execute(playerTest, new string[] { "look", "at", "sword", "in
                Is.EqualTo(unknown));
        }

        //! Tests all error conditions
        public void TestInvalidLook(string look, string result)
        {
            Assert.Multiple(() => {
                Assert.That(lookTest.Execute(playerTest, new string[] {"aaaaa"}),
                    Is.EqualTo(badLength));
                Assert.That(lookTest.Execute(playerTest, new string[] { "search", "at", "sword
                    Is.EqualTo(badLook));
                Assert.That(lookTest.Execute(playerTest, new string[] { "look", "for", "sword"
                    Is.EqualTo(badAt));
                Assert.That(lookTest.Execute(playerTest, new string[] { "look", "for", "sword"
                    Is.EqualTo(badIn));
            }); //? can i use testcases here?
        }
    }
}
```