Swinburne University of Technology

COS20007 Object Oriented Programming

Case Study - Iteration 6 - Locations

PDF generated at 18:12 on Saturday 21st October, 2023

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace SwinAdventure
{
    public class Location : GameObject, IHaveInventory
    {
        private Inventory _inventory;

        public Location(string[] ids, string name, string description)
            : base(ids, name, description)
        {
            _inventory = new Inventory();
        }

        public GameObject Locate(string id)
        {
            if (AreYou(id))
            {
                return this;
            }

            GameObject item = _inventory.Fetch(id);
            if (item != null)
            {
                return item;
            }

            return _inventory.Locate(id);
        }

        public Inventory Inventory
        {
            get { return _inventory; }
        }
    }
}

```

```
using NUnit.Framework;
using SwinAdventure;

namespace SwinAdventureTest
{
    [TestFixture]
    public class LocationTests
    {
        [Test]
        public void TestLocationIdentification()
        {
            Location location = new Location(new string[] { "forest" }, "Dark Forest", "A mysterious and dark forest.");

            Assert.IsTrue(location.AreYou("forest"));
            Assert.IsFalse(location.AreYou("cave"));
        }

        [Test]
        public void TestLocationLocateItems()
        {
            Location location = new Location(new string[] { "forest" }, "Dark Forest", "A mysterious and dark forest.");
            Item item = new Item(new string[] { "flower" }, "Red Flower", "A beautiful red flower.");
            location.Inventory.Put(item);

            Assert.AreSame(item, location.Locate("flower"));
        }
        [Test]
        public void TestPlayerLocateItemsInLocation()
        {
            Player player = new Player("Alice", "Adventurer");
            Location location = new Location(new string[] { "castle" }, "Castle", "A majestic castle.");
            Item item = new Item(new string[] { "crown" }, "Royal Crown", "A shiny royal crown.");
            location.Inventory.Put(item);
            player.Location = location;

            Assert.AreSame(item, player.Locate("crown"));
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace SwinAdventure
{
    public class Player : GameObject, IHaveInventory
    {
        private Inventory _inventory;

        private Location _location;

        public Location Location
        {
            get { return _location; }
            set { _location = value; }
        }

        public Player(string name, string desc) : base(new string[] { "me", "inventory" }, name, desc)
        {
            _inventory = new Inventory();
        }

        public GameObject? Locate(string id)
        {
            if (AreYou(id))
            {
                return this;
            }
            else if (_inventory.HasItem(id))
            {
                return _inventory.Fetch(id);
            }
            else
            {
                if (Location != null)
                {
                    return Location.Locate(id);
                }
                else
                {
                    return null;
                }
            }
        }

        public override string FullDescription
        {
            get
            {
                string playerDescription = $"You are {Name}, {base.FullDescription}. You are carrying:\n{_inventory.ItemList}";
```

```
                return playerDescription;
            }
        }

        public Inventory Inventory
        {
            get { return _inventory; }
        }
    }
}

```

```
using System;
using NUnit.Framework;
using SwinAdventure;

namespace SwinAdventureTest
{
    [TestFixture]
    public class PlayerTests
    {
        [Test]
        public void PlayerIsIdentifiable()
        {
            Player player = new Player("Fred", "the mighty programmer");

            Assert.IsTrue(player.AreYou("me"), "Player should respond to 'me'");
            Assert.IsTrue(player.AreYou("inventory"), "Player should respond to 'inventory'");
            Assert.IsFalse(player.AreYou("player"), "Player should not respond to 'player'");
        }

        [Test]
        public void PlayerLocatesItemsInInventory()
        {
            Player player = new Player("Fred", "the mighty programmer");
            Item item = new Item(new string[] { "sword" }, "bronze sword", "This is a mighty fine sword.");
            player.Inventory.Put(item);

            GameObject locatedItem = player.Locate("sword");

            Assert.IsNotNull(locatedItem, "Player should locate 'sword'");
            Assert.IsTrue(locatedItem.AreYou("sword"), "Located item should be 'sword'");
            Assert.IsTrue(player.Inventory.HasItem("sword"), "Item should remain in player's inventory");
        }

        [Test]
        public void PlayerLocatesItself()
        {
            Player player = new Player("Fred", "the mighty programmer");

            GameObject locatedPlayer1 = player.Locate("me");
            GameObject locatedPlayer2 = player.Locate("inventory");

            Assert.IsNotNull(locatedPlayer1, "Player should locate 'me'");
            Assert.IsTrue(locatedPlayer1.AreYou("me"), "Located object should be 'me'");
            Assert.IsNotNull(locatedPlayer2, "Player should locate 'inventory'");
            Assert.IsTrue(locatedPlayer2.AreYou("inventory"), "Located object should be 'inventory'");
        }
```

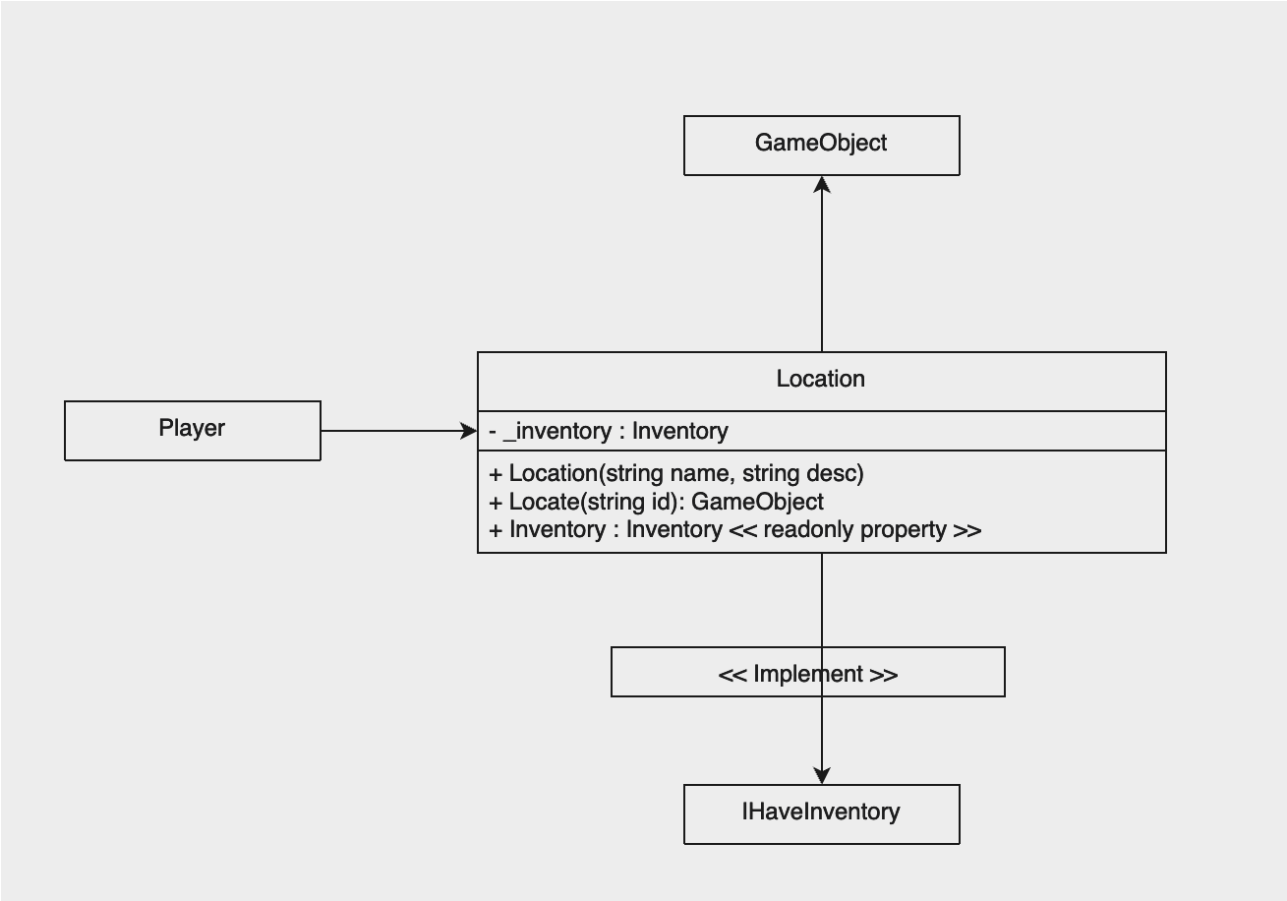
```

        [Test]
        public void PlayerLocatesNothing()
        {
            Player player = new Player("Fred", "the mighty programmer");

            GameObject locatedObject = player.Locate("axe");

            Assert.IsNull(locatedObject, "Player should not locate 'axe'");
        }

        [Test]
        public void PlayerFullDescriptionContainsItems()
        {
            Player player = new Player("Fred", "the mighty programmer");
            Item item1 = new Item(new string[] { "shovel" }, "a shovel", "A gardening shovel.");
            Item item2 = new Item(new string[] { "sword" }, "a sword", "A sharp sword.");
            player.Inventory.Put(item1);
            player.Inventory.Put(item2);

            string fullDescription = player.FullDescription;

            Assert.IsTrue(fullDescription.Contains("You are Fred, the mighty programmer."), "FullDescription should contain player's name and description");
            Assert.IsTrue(fullDescription.Contains("You are carrying:"), "FullDescription should contain 'You are carrying:'");
            Assert.IsTrue(fullDescription.Contains("a shovel (shovel)"), "FullDescription should contain 'a shovel'");
            Assert.IsTrue(fullDescription.Contains("a sword (sword)"), "FullDescription should contain 'a sword'");
        }
    }
}
```

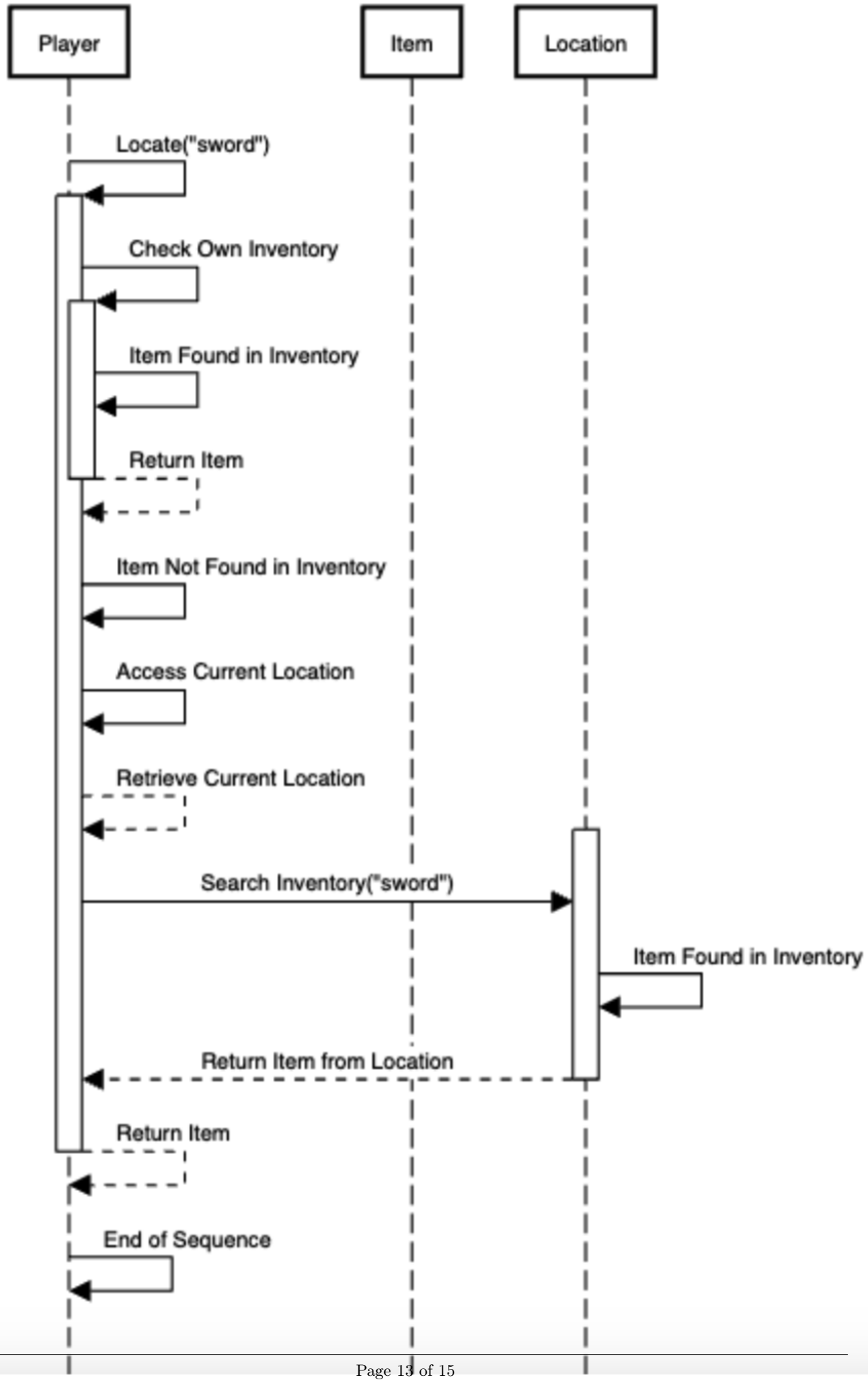
```
using System;
using System.Collections.Generic;
using System.Linq;

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

        public override string Execute(Player player, string[] text)
        {
            if (text.Length == 1)
            {
                return player.FullDescription;
            }
            else if (text.Length == 3 && text[1] == "at")
            {
                string itemToLookAt = text[2];
                GameObject item = player.Locate(itemToLookAt);

                if (item != null)
                {
                    return item.FullDescription;
                }
                else
                {
                    return $"I cannot find the {itemToLookAt}.";
                }
            }
            else if (text.Length == 5 && text[1] == "at" && text[3] == "in")
            {
                string itemToLookAt = text[2];
                string containerId = text[4];

                GameObject container = player.Locate(containerId);

                if (container != null && container is IHaveInventory)
                {
                    IHaveInventory containerWithInventory = container as IHaveInventory;
                    GameObject item = containerWithInventory.Locate(itemToLookAt);

                    if (item != null)
                    {
                        return item.FullDescription;
                    }
                    else
                    {
                        return $"I cannot find the {itemToLookAt} in the {container.Name}.";
                    }
                }
```

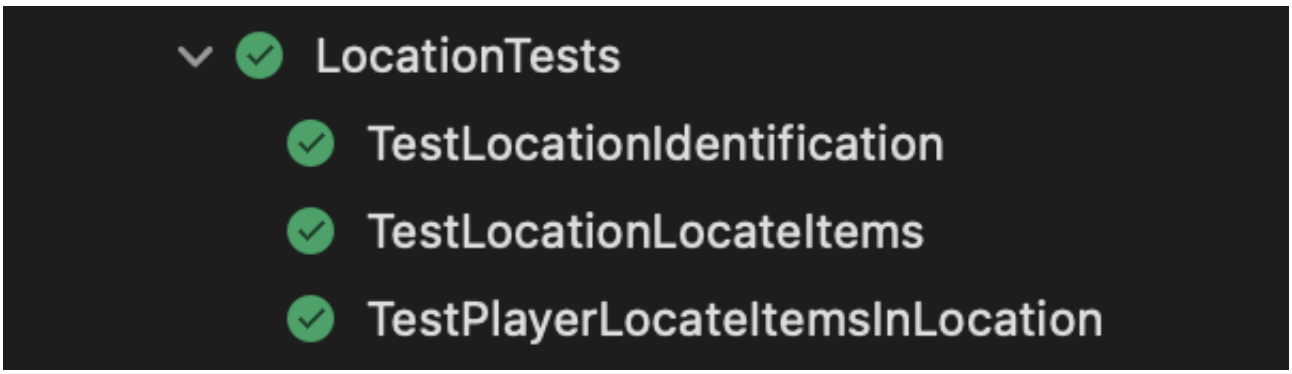
```
                else
                {
                    return $"I cannot find the {containerId}.";
                }
            }
            else if (text.Length == 2 && text[1] == "location")
            {
                return player.Location.FullDescription;
            }
            else
            {
                return "Look at what?";
            }
        }
    }
}
```

```
using System;
using NUnit.Framework;
using SwinAdventure;

namespace SwinAdventureTest
{
    [TestFixture]
    public class LookCommandTests
    {
        [Test]
        public void LookAtMe()
        {
            Player player = new Player("Fred", "the mighty programmer");
            LookCommand lookCmd = new LookCommand();

            string result = lookCmd.Execute(player, new string[] { "look", "at", "me" });

            Assert.AreEqual(player.FullDescription, result);
        }

        [Test]
        public void LookAtGemInInventory()
        {
            Player player = new Player("Fred", "the mighty programmer");
            Item gem = new Item(new string[] { "gem" }, "shiny gem", "A beautiful gemstone.");
            player.Inventory.Put(gem);
            LookCommand lookCmd = new LookCommand();

            string result = lookCmd.Execute(player, new string[] { "look", "at", "gem" });

            Assert.AreEqual(gem.FullDescription, result);
        }

        [Test]
        public void LookAtUnk()
        {
            Player player = new Player("Fred", "the mighty programmer");
            LookCommand lookCmd = new LookCommand();

            string result = lookCmd.Execute(player, new string[] { "look", "at", "unknown" });

            Assert.AreEqual("I cannot find the unknown.", result);
        }

        [Test]
        public void LookAtGemInMe()
        {
            Player player = new Player("Fred", "the mighty programmer");
            Item gem = new Item(new string[] { "gem" }, "shiny gem", "A beautiful gemstone.");
```

```
            player.Inventory.Put(gem);
            LookCommand lookCmd = new LookCommand();

            string result = lookCmd.Execute(player, new string[] { "look", "at", "gem", "in", "me" });

            Assert.AreEqual(gem.FullDescription, result);
        }

        [Test]
        public void LookAtGemInBagInInventory()
        {
            Player player = new Player("Fred", "the mighty programmer");
            Item gem = new Item(new string[] { "gem" }, "shiny gem", "A beautiful gemstone.");
            Bag bag = new Bag(new string[] { "bag" }, "small bag", "A small bag.");
            bag.Inventory.Put(gem);
            player.Inventory.Put(bag);
            LookCommand lookCmd = new LookCommand();

            string result = lookCmd.Execute(player, new string[] { "look", "at", "gem", "in", "bag" });

            Assert.AreEqual(gem.FullDescription, result);
        }

        [Test]
        public void LookAtGemInNoBag()
        {
            Player player = new Player("Fred", "the mighty programmer");
            Item gem = new Item(new string[] { "gem" }, "shiny gem", "A beautiful gemstone.");
            LookCommand lookCmd = new LookCommand();

            string result = lookCmd.Execute(player, new string[] { "look", "at", "gem", "in", "bag" });

            Assert.AreEqual("I cannot find the bag.", result);
        }

        [Test]
        public void InvalidLookOptions()
        {
            Player player = new Player("Fred", "the mighty programmer");
            LookCommand lookCmd = new LookCommand();

            string result1 = lookCmd.Execute(player, new string[] { "look", "around" });
        }
    }
}
```


GameObject
Location
- _inventory : Inventory
+ Location(string name, string desc)
+ Locate(string id): GameObject
+ Inventory : Inventory << readonly property >>
Player
<< Implement >>
IHaveInventory


Player
Item
Location
Locate("sword")
Check Own Inventory
Item Found in Inventory
Return Item
Item Not Found in Inventory
Access Current Location
Retrieve Current Location
Search Inventory("sword")
Item Found in Inventory
Return Item from Location
Return Item
End of Sequence


LocationTests
TestLocationIdentification
TestLocationLocateItems
TestPlayerLocateItemsInLocation

```
Welcome to SwinAdventure!
Enter your player's name: Phan Vu
Enter a description for your player: Strong AF!
Enter a command or type 'exit' to quit: add sword to bag
bronze sword has been added to small bag.
Enter a command or type 'exit' to quit: add potion to bag
healing potion has been added to small bag.
Enter a command or type 'exit' to quit: add bag to location
Enter a command or type 'exit' to quit: look bag
In the small bag you can see:
        a shiny gem (gem)
        a bronze sword (sword)
        a healing potion (potion)

Enter a command or type 'exit' to quit: look location
A mysterious and dark forest.
Items in the location:
        a bronze sword (sword)
        a healing potion (potion)
        a shiny gem (gem)
        a small bag (bag)


Enter a command or type 'exit' to quit:
```