```
using Iteration4;
using NUnit.Framework;

namespace UTLookCommand
{
    public class Tests
    {
        private Player _player;
        private Item gem, spade;
        private Bag bag;
        private LookCommand lookCommand;

        [SetUp]
        public void Setup()
        {
            _player = new Player("Player1", "first player");
            gem = new Item(new string[] { "gem", "a gem" }, "purple gem", 
              "big purple gem");
            spade = new Item(new string[] { "spade" }, "purple spade", 
              "big purple spade");

            lookCommand = new LookCommand();
            bag = new Bag(new string[] { "testBag" }, "a bag", "contains 
              items");
        }

        [Test]
        public void LookAtMeTest()
        {
            _player.Inventory.Put(gem);
            var expectedOutcome = _player.FullDescription;
            string desc = lookCommand.Execute(_player, new string[]
              { "look", "at", "inventory" });
            Assert.That(desc, Is.EqualTo(expectedOutcome));
        }

        [Test]
        public void LookAtGemTest()
        {
            _player.Inventory.Put(gem);
            var expectedOutcome = gem.FullDescription;
            var result = lookCommand.Execute(_player, new string[]
              { "look", "at", "gem" });
            Assert.That(result, Is.EqualTo(expectedOutcome));
        }

        [Test]
        public void LookAtUnkTest()
        {
            var expectedOutcome = "I can't find the gem";
            var result = lookCommand.Execute(_player, new string[]
              { "look", "at", "gem" });
            Assert.That(result, Is.EqualTo(expectedOutcome));
        }

        [Test]
        public void LookAtGemInMeTest()
        {
            _player.Inventory.Put(gem);
            var expectedOutcome = gem.FullDescription;
            var result = lookCommand.Execute(_player, new string[]
              { "look", "at", "gem", "in", "me" });
            Assert.That(result, Is.EqualTo(expectedOutcome));
        }

        [Test]
        public void LookAtGeminBagTest()
        {
            _player.Inventory.Put(gem);
            var expectedOutcome = gem.FullDescription;
            var result = lookCommand.Execute(_player, new string[]
              { "look", "at", "gem", "in", "Inventory" });
            Assert.That(result, Is.EqualTo(expectedOutcome));
        }

        [Test]
        public void LookAtGemInNoBagTest()
        {
            var expectedOutcome = "I can't find the bag";
            var result = lookCommand.Execute(_player, new string[]
              { "look", "at", "bag", "in", "me" });
            Assert.That(result, Is.EqualTo(expectedOutcome));
        }

        [Test]
        public void LookAtNoGemInBagTest()
        {
            var expectedOutcome = "I can't find the gem";
            var result = lookCommand.Execute(_player, new string[]
              { "look", "at", "gem", "in", "me" });
            Assert.That(result, Is.EqualTo(expectedOutcome));
        }

        [Test]
        public void InvalidLookTest()
        {
            var result0 = lookCommand.Execute(_player, new string[]
              { "look", "there" });
            Assert.That(result0, Is.EqualTo("I don't know how to look like 
              that"));

            var result1 = lookCommand.Execute(_player, new string[]        
              { "there", "it", "is" });
            Assert.That(result1, Is.EqualTo("Error in look input"));

            var result2 = lookCommand.Execute(_player, new string[]        
              { "look", "over", "there" });
            Assert.That(result2, Is.EqualTo("What do you want to look      
              at?"));

            var result3 = lookCommand.Execute(_player, new string[]        
              { "look", "at", "gem", "over", "there" });
            Assert.That(result3, Is.EqualTo("What do you want to look      
              in?"));
        }
    }
}
```