```
namespace IdentifiableObject
{
    public class Tests
    {
        private Player _player;
        private Player _player_no_bag;
        private Item _gem;
        private Bag _bag;
        private Command _look;

        [SetUp]
        public void Setup()
        {
            _look = new LookCommand();
            _player = new Player("Duc Thang", "Student");
            _player_no_bag = new Player("player", "participant");
            _gem = new Item(new string[] { "gem" }, "a gem", "This is a
              gem");
            _bag = new Bag(new string[] { "bag" }, "Thang's bag",
              "student");
            _player.Inventory.Put(_bag);
        }

        [Test]
        public void TestLookAtMe()
        {
            string look_execution = _look.Execute(_player, new string[]
              { "look", "at", "inventory" });
            string output = _player.FullDescription;
            Assert.AreEqual(look_execution, output);
        }

        [Test]
        public void TestLookAtGem()
        {
            _player.Inventory.Put(_gem);
            string look_execution = _look.Execute(_player, new string[]
              {"look", "at", "gem"});
            string output = _gem.FullDescription;
            Assert.AreEqual(look_execution, output);
        }

        [Test]
        public void TestLookAtUnk()
        {
            string look_execution = _look.Execute(_player, new string[]
              { "look", "at", "gem" });
            string output = "I can't find the gem";
            Assert.AreEqual(look_execution, output);
        }

        [Test]
        public void TestLookAtGemInMe()
        {
            _player.Inventory.Put(_gem);
            string look_execution = _look.Execute(_player, new string[]
              { "look", "at", "gem", "in", "me" });
            string output = _gem.FullDescription;
            Assert.AreEqual(look_execution, output);
        }

        [Test]
        public void TestLookAtGemInBag()
        {
            _bag.Inventory.Put(_gem);
            string look_execution = _look.Execute(_player, new string[]
              { "look", "at", "gem", "in", "bag" });
            string output = _gem.FullDescription;
            Assert.AreEqual(look_execution, output);
        }

        [Test]
        public void TestLookAtGemInNoBag()
        {
            string look_execution = _look.Execute(_player_no_bag, new
              string[] { "look", "at", "bag" });
            string output = "I can't find the bag";
            Assert.AreEqual(look_execution, output);
        }

        [Test]
        public void TestLookAtNoGemInBag()
        {
            string look_execution = _look.Execute(_player, new string[]
              { "look", "at", "gem", "in", "bag" });
            string output = "I can't find the gem";
            Assert.AreEqual(look_execution, output);
        }

        [Test]
        public void TestInvalidLook()
        {
            Assert.AreEqual(_look.Execute(_player, new string[] { "look",
              "around" }), "I don't know how to look like that");
            Assert.AreEqual(_look.Execute(_player, new string[] { "hello",
              "104776473" }), "I don't know how to look like that");
            Assert.AreEqual(_look.Execute(_player, new string[] { "look",
              "at", "Nguyen Duc Thang" }), "I can't find the Nguyen Duc
```

```
                    Thang");
            }
        }
    }
```