6.1. All cs. Program files are included in Submission.

LookCommand Test Code:

```
        [Test]
        public void LookAtMe()
        {
            Player player = new Player("Tan", "A player");
            player.Inventory.Put(item1);
            player.Inventory.Put(item2);
            LookCommand LookCommand = new LookCommand();

            string expectedDescription = "You are Tan A player\nYou are
carrying:\nsword (sword)\nshield (shield)\n";
            string testDescription = LookCommand.Execute(player, new string[] {
"look", "at", "me" });
            Assert.That(testDescription, Is.EqualTo(expectedDescription));

        }

        [Test]
        public void LookAtGem()
        {
            Player player = new Player("Tan", "A player");
            player.Inventory.Put(item4);
            LookCommand LookCommand = new LookCommand();

            string expectedDescription = "a gem";
            string testDescription = LookCommand.Execute(player, new string[] {
"look", "at", "gem" });
            Assert.That(testDescription, Is.EqualTo(expectedDescription));
        }

        [Test]
        public void LookAtUnk()
        {
            Player player = new Player("Tan", "A player");
            LookCommand LookCommand = new LookCommand();

            string expectedDescription = "I can't find the gem in the Tan";
            string testDescription = LookCommand.Execute(player, new string[] {
"look", "at", "gem" });
            Assert.That(testDescription, Is.EqualTo(expectedDescription));
        }

        [Test]
        public void LookAtGemInBag()
        {
            Player player = new Player("Tan", "A player");
            Bag backpack = new Bag(new string[] { "backpack" }, "backpack", "a
backpack");
            player.Inventory.Put(backpack);
            backpack.Inventory.Put(item4);
            LookCommand LookCommand = new LookCommand();

            string expectedDescription = "a gem";
            string testDescription = LookCommand.Execute(player, new string[] {
"look", "at", "gem", "in", "backpack" });
            Assert.That(testDescription, Is.EqualTo(expectedDescription));
        }

        [Test]
```

```
        public void LookAtGemInNoBag()
        {
            Player player = new Player("Tan", "A player");
            LookCommand LookCommand = new LookCommand();

            string expectedDescription = "I can't find the backpack";
            string testDescription = LookCommand.Execute(player, new string[] {
"look", "at", "gem", "in", "backpack" });
            Assert.That(testDescription, Is.EqualTo(expectedDescription));
        }

        [Test]
        public void LookAtNoGemInBag()
        {
            Player player = new Player("Tan", "A player");
            Bag backpack = new Bag(new string[] { "backpack" }, "backpack", "a
backpack");
            player.Inventory.Put(backpack);
            LookCommand LookCommand = new LookCommand();

            string expectedDescription = "I can't find the gem in the backpack";
            string testDescription = LookCommand.Execute(player, new string[] {
"look", "at", "gem", "in", "backpack" });
            Assert.That(testDescription, Is.EqualTo(expectedDescription));
        }

        [Test]
        public void InvalidLookCommand()
        {
            Player player = new Player("Tan", "A player");
            LookCommand LookCommand = new LookCommand();

            string expectedDescription = "I don't know how to look like that";

            //only 2 arguments
            string testDescription = LookCommand.Execute(player, new string[] {
"look", "at" });
            Assert.That(testDescription, Is.EqualTo(expectedDescription));

            //4 arguments
            string testDescription2 = LookCommand.Execute(player, new string[] {
"look", "at", "gem", "in" });
            Assert.That(testDescription2, Is.EqualTo(expectedDescription));

            //5 arguments but the 4th argument is not "in"
            string testDescription3 = LookCommand.Execute(player, new string[] {
"look", "at", "a", "at", "b" });
            string expectedDescription2 = "What do you want to look in?";
            Assert.That(testDescription3, Is.EqualTo(expectedDescription2));

            //5 arguments but the 2nd argument is not "at"
            string testDescription4 = LookCommand.Execute(player, new string[] {
"look", "in", "a", "in", "b" });
            string expectedDescription3 = "What do you want to look at?";
            Assert.That(testDescription4, Is.EqualTo(expectedDescription3));
        }
```

Test Results

Test Explorer

25 25 0

Test run finished: 25 Tests (25 Passed, 0 Failed, 0 Skipped) run in 149 ms

Test	Duration	Traits	Error...
TestQueue (25)	5 ms		
TestQueue (25)	5 ms		
Tests (25)	5 ms		
BagFullDescription	5 ms		
BagInBag	< 1 ms		
BagLocate	< 1 ms		
BagLocateNothing	< 1 ms		
BagLocatesItself	< 1 ms		
FetchItem	< 1 ms		
FindItem	< 1 ms		
FullDescription	< 1 ms		
InvalidLookCommand	< 1 ms		
ItemIdentifiable	< 1 ms		
ItemList	< 1 ms		
LookAtGem	< 1 ms		
LookAtGemInBag	< 1 ms		
LookAtGemInNoBag	< 1 ms		
LookAtMe	< 1 ms		
LookAtNoGemInBag	< 1 ms		
LookAtUnk	< 1 ms		
NoItem	< 1 ms		
PlayerFullDescription	< 1 ms		
PlayerIdentifiable	< 1 ms		
PlayerLocate	< 1 ms		
PlayerLocateItself	< 1 ms		
PlayerLocateNothing	< 1 ms		
ShortDescription	< 1 ms		
TakeItem	< 1 ms		