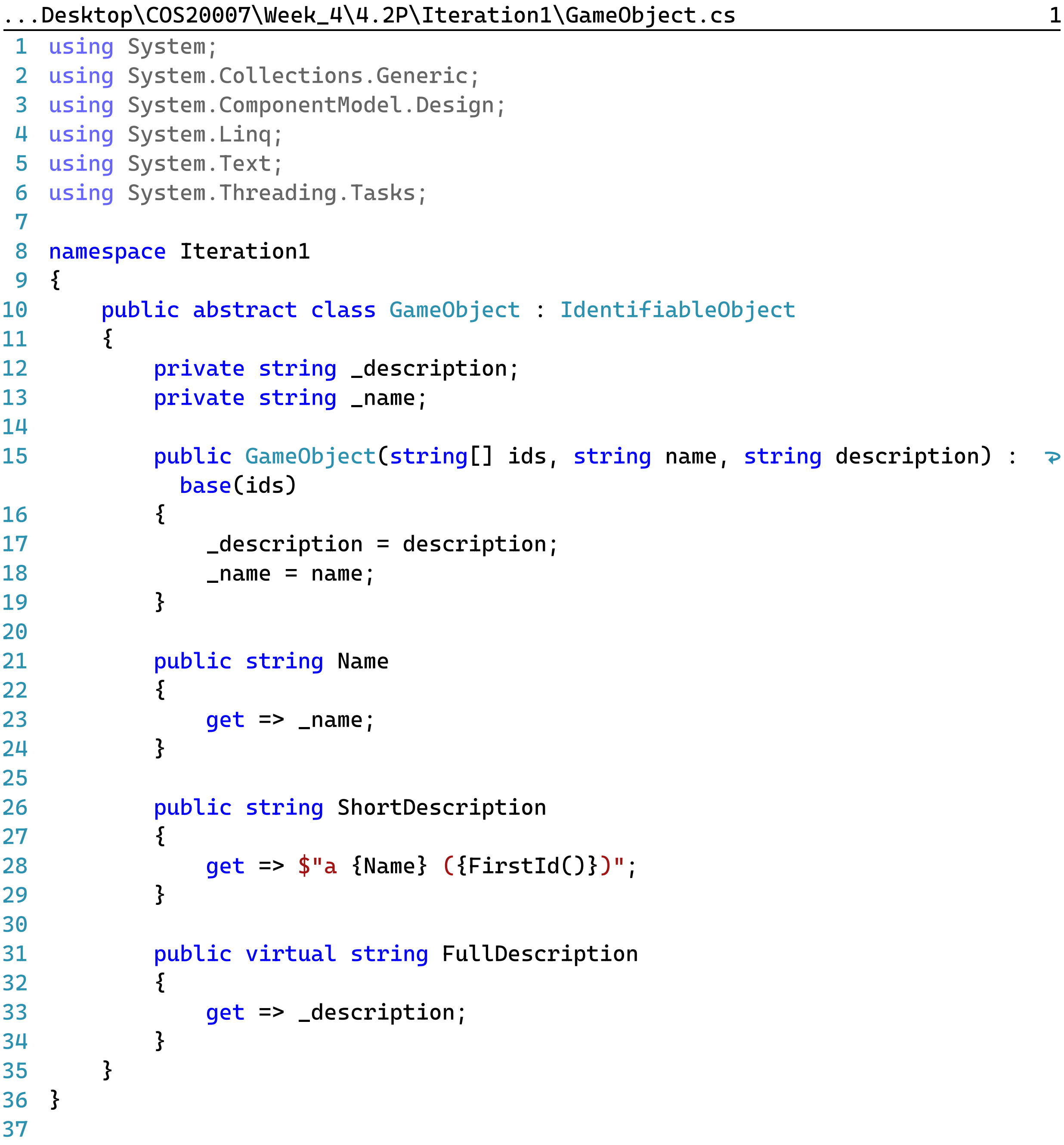
GameObject.cs



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

namespace Iteration1
{
    public abstract class GameObject : IdentifiableObject
    {
        private string _description;
        private string _name;

        public GameObject(string[] ids, string name, string description) : base(ids)
        {
            _description = description;
            _name = name;
        }

        public string Name
        {
            get => _name;
        }

        public string ShortDescription
        {
            get => $"a {Name} ({FirstId()})";
        }

        public virtual string FullDescription
        {
            get => _description;
        }
    }
}
```

IdentifiableObject.cs

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

namespace Iteration1
{
    public class IdentifiableObject
    {
        private List<string> _identifiers = new List<string>();

        public IdentifiableObject(string[] idents)
        {
            for (int i = 0; i < idents.Length; i++)
            {
                _identifiers.Add(idents[i].ToLower());
            }
        }
        public bool AreYou(string id)
        {
            if (_identifiers.Contains(id.ToLower()))
            {
                return true;
            }
            return false;
        }
        public string FirstId()
        {
            if (_identifiers.Any())
            {
                return _identifiers.First();
            }
            return "";
        }
        public void AddIdentifier(string id)
        {
            _identifiers.Add(id.ToLower());
        }
    }
}

```

Inventory.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.IO.Pipes;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Iteration1
{
    public class Inventory
    {
        private List<Item> _items;

        public Inventory()
        {
            _items = new List<Item>();
        }

        public bool HasItem(string id)
        {
            foreach (Item itm in  _items)
            {
                if (itm.AreYou(id))
                {
                    return true;
                }
            }
            return false;
        }

        public void Put(Item itm)
        {
            _items.Add(itm);
        }

        public Item Fetch(string id)
        {
            foreach (Item itm in _items)
            {
                if (itm.AreYou(id))
                {
                    return itm;
                }
            }
            return null;
        }

        public Item Take(string id)
        {
            Item a = Fetch(id);
            _items.Remove(a);
            return a;
        }

        public string ItemList
        {
            get
            {
                string list = "";
                foreach (Item itm in _items)
                {
                    list += "    " + itm.ShortDescription + "\n";
                }
                return list;
            }
        }

    }
}

```

Item.cs

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

namespace Iteration1
{
    public class Item : GameObject
    {
        public Item(string[] idents, string name, string description) : base(idents, name, description)
        {
        }
    }
}

```

Player.cs

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection.Metadata.Ecma335;
using System.Text;
using System.Threading.Tasks;

namespace Iteration1
{
    public class Player : GameObject
    {
        private Inventory _inventory;

        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);
            }
            return null;
        }

        public override string FullDescription
        {
            get
            {
                string fulldesc = "";
                fulldesc += $"You are {Name}, {base.FullDescription}\n";
                fulldesc += "You are carrying\n";
                fulldesc += $"{_inventory.ItemList}";
                return fulldesc;
            }
        }

        public Inventory Inventory
        {
            get => _inventory;
        }
    }
}

```

Program.cs

```
namespace Iteration1
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello, World!");
        }
    }
}

```

InventoryUnitTests.cs

```csharp
using Iteration1;

namespace InventoryUnitTests
{
    public class Tests
    {
        Inventory ivt;
        Item sword;
        Item ak47;

        [SetUp]
        public void Setup()
        {
            ivt = new Inventory();
            sword = new Item(new string[] { "sword", "melee" }, "bronze sword", "Melee weapon. High damage.");
            ak47 = new Item(new string[] { "ak47", "gun" }, "ak47", "Gun. High Damage.");
            ivt.Put(sword);
            ivt.Put(ak47);
        }

        [Test]
        public void TestFindItem()
        {
            Assert.That(ivt.HasItem("sword"), Is.EqualTo(true));
            Assert.Pass();
        }

        [Test]
        public void TestNoItemFind()
        {
            Assert.That(ivt.HasItem("machine gun"), Is.EqualTo(false));
            Assert.Pass();
        }

        [Test]
        public void TestFetchItem()
        {
            Assert.That(ivt.Fetch("sword"), Is.EqualTo(sword));
            Assert.That(ivt.HasItem("sword"), Is.EqualTo(true));
            Assert.Pass();
        }

        [Test]
        public void TestTakeItem()
        {
            Assert.That(ivt.Take("ak47"), Is.EqualTo(ak47));
            Assert.That(ivt.HasItem("ak47"), Is.EqualTo(false));
            Assert.Pass();
        }

        [Test]
        public void TestItemList()
        {
            Assert.That(ivt.ItemList, Is.EqualTo("    a bronze sword (sword)\n    a ak47 (ak47)\n"));
            Assert.Pass();
        }
    }
}
```

ItemUnitTests

```
using Iteration1;

namespace ItemUnitTests
{
    public class Tests
    {
        Item itm;
        [SetUp]
        public void Setup()
        {
            itm = new Item(new string[] {"sword", "great"}, "bronze sword", "Melee weapon. High damage.");
        }

        [Test]
        public void TestItemIsIdentifiable()
        {
            Assert.That(itm.AreYou("sword"), Is.EqualTo(true));
            Assert.Pass();
        }

        [Test]
        public void TestShortDescription()
        {
            Assert.That(itm.ShortDescription, Is.EqualTo("a bronze sword (sword)"));
            Assert.Pass();
        }

        [Test]
        public void TestFullDescription()
        {
            Assert.That(itm.FullDescription, Is.EqualTo("Melee weapon. High damage."));
            Assert.Pass();
        }
    }
}
```

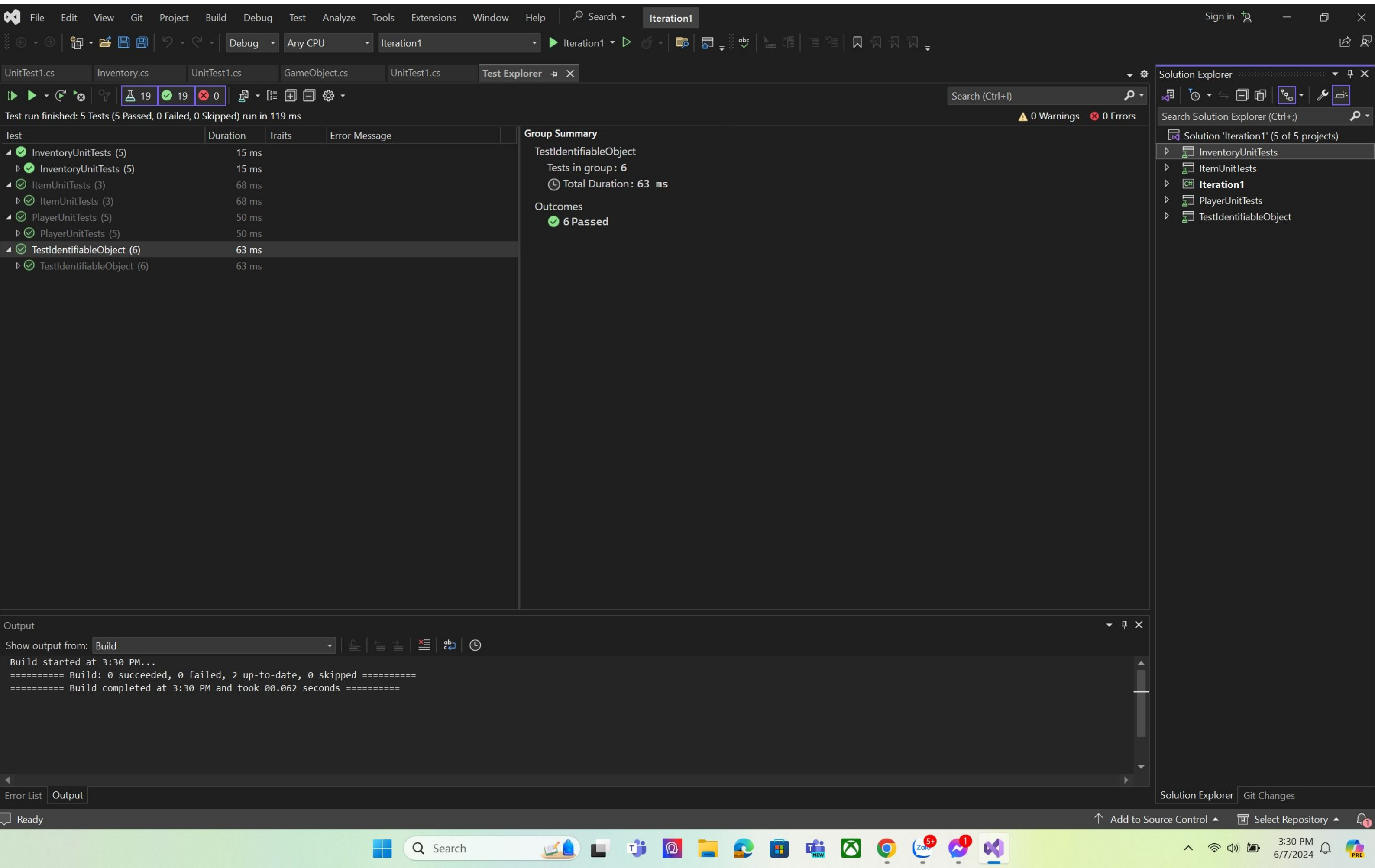
PlayerUnitTests

```
using Iteration1;

namespace PlayerUnitTests
{
    public class Tests
    {
        Player player;
        Item sword;
        Item ak47;

        [SetUp]
        public void Setup()
        {
            player = new Player("Chien", "A boy with high curiosity.");
            sword = new Item(new string[] { "sword", "melee" }, "bronze sword", "Melee weapon. High damage.");
            ak47 = new Item(new string[] { "ak47", "gun" }, "ak47", "Gun. High Damage.");
            player.Inventory.Put(sword);
            player.Inventory.Put(ak47);
        }

        [Test]
        public void TestPlayerIdentifiable()
        {
            Assert.That(player.AreYou("inventory"), Is.EqualTo(true));
            Assert.Pass();
        }

        [Test]
        public void TestPlayerLocatesItem()
        {
            Assert.That(player.Locate("sword"), Is.EqualTo(sword));
            Assert.That(player.Inventory.HasItem("sword"), Is.EqualTo(true));
            Assert.Pass();
        }

        [Test]
        public void TestPlayerLocatesItself()
        {
            Assert.That(player.Locate("inventory"), Is.EqualTo(player));
            Assert.That(player.Locate("me"), Is.EqualTo(player));
            Assert.Pass();
        }

        [Test]
        public void TestPlayerLocatesNothing()
        {
            Assert.That(player.Locate("machine gun"), Is.EqualTo(null));
            Assert.Pass();
        }

        [Test]
        public void TestPlayerFullDescription()
        {
            Assert.That(player.FullDescription, Is.EqualTo("You are Chien, A boy with high curiosity.\nYou are carrying\n    a bronze sword (sword)\n    a ak47 (ak47)\n"));
            Assert.Pass();
        }
    }
}
```

TestIdentifiableObjects

```
using Iteration1;

namespace TestIdentifiableObject
{
    public class Tests
    {
        private IdentifiableObject myObject;

        [SetUp]
        public void SetUp()
        {
            myObject = new IdentifiableObject(new string[] { "fred", "bob" });
        }

        [Test]
        public void TestAreYou()
        {
            Assert.That(myObject.AreYou("bob"), Is.EqualTo(true));
            Assert.That(myObject.AreYou("fred"), Is.EqualTo(true));

            Assert.Pass();
        }

        [Test]
        public void TestNotAreYou()
        {
            Assert.That(myObject.AreYou("wilma"), Is.EqualTo(false));
            Assert.That(myObject.AreYou("boby"), Is.EqualTo(false));

            Assert.Pass();
        }

        [Test]
        public void TestCaseSentive()
        {
            Assert.That(myObject.AreYou("FRED"), Is.EqualTo(true));
            Assert.That(myObject.AreYou("bOB"), Is.EqualTo(true));

            Assert.Pass();
        }

        [Test]
        public void TestFirstId()
        {
            Assert.That(myObject.FirstId(), Is.EqualTo("fred"));

            Assert.Pass();
        }

        [Test]
        public void TestFirstIdWithoutIDs()
        {
            IdentifiableObject myObject = new IdentifiableObject(new string [] { });

            Assert.That(myObject.FirstId(), Is.EqualTo(""));

            Assert.Pass();
        }

        [Test]
        public void TestAddId()
        {
            myObject.AddIdentifier("wilma");

            Assert.That(myObject.AreYou("fred"), Is.EqualTo(true));
            Assert.That(myObject.AreYou("bob"), Is.EqualTo(true));
            Assert.That(myObject.AreYou("wilma"), Is.EqualTo(true));

            Assert.Pass();
        }
    }
}
```

Screenshot of Test Passing



Test run finished: 5 Tests (5 Passed, 0 Failed, 0 Skipped) run in 119 ms

Test	Duration	Traits	Error Message
InventoryUnitTests (5)	15 ms		
InventoryUnitTests (5)	15 ms		
ItemUnitTests (3)	68 ms		
ItemUnitTests (3)	68 ms		
PlayerUnitTests (5)	50 ms		
PlayerUnitTests (5)	50 ms		
TestIdentifiableObject (6)	63 ms		
TestIdentifiableObject (6)	63 ms		

Group Summary

TestIdentifiableObject

Tests in group: 6

Total Duration: 63 ms

Outcomes

✓ 6 Passed

Output

Show output from: Build

```
Build started at 3:30 PM...
========== Build: 0 succeeded, 0 failed, 2 up-to-date, 0 skipped ==========
========== Build completed at 3:30 PM and took 00.062 seconds ==========
```