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

namespace SwinAdventure
{
    public class IdentifiableObject
    {
        private List<string> _identifiers;

        public IdentifiableObject(string[] idents)
        {
            _identifiers = new List<string>();
            foreach (string id in idents)
            {
                _identifiers.Add(id.ToLower());
            }
        }

        public bool AreYou(string id)
        {
            return _identifiers.Contains(id.ToLower());
        }

        public string FirstId
        {
            get
            {
                if (_identifiers.Count > 0)
                {
                    return _identifiers[0];
                }
                else
                {
                    return "";
                }
            }
        }

        public void AddIdentifier(string id)
        {
            _identifiers.Add(id.ToLower());
        }
    }
}

```

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

namespace SwinAdventure
{
    public class GameObject : IdentifiableObject
    {
        private string _name;
        private string _description;
        private string[] _ids;

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

        public string Name
        {
            get
            {
                return _name;
            }
        }

        public string ShortDescription
        {
            get
            {
                return _name + " (" + _ids[0] + ")";
            }
        }

        public virtual string FullDescription
        {
            get
            {
                return _description;
            }
        }
    }
}

```

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

namespace SwinAdventure
{
    public class Inventory
    {
        List<Item> _items;

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

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

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

        public Item? Take(string id)
        {
            Item? t = null;
            foreach (Item i in _items)
            {
                if (i.AreYou(id))
                {
                    t = i;
                    _items.Remove(i);
                    return t;
                }
            }
            return t;
        }

        public Item? Fetch(string id)
        {
            Item? t = null;
            foreach (Item i in _items)
            {
                if (i.AreYou(id))
                {
                    return i;
                }
            }
            return t;
        }

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

```

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

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

    }
}

```

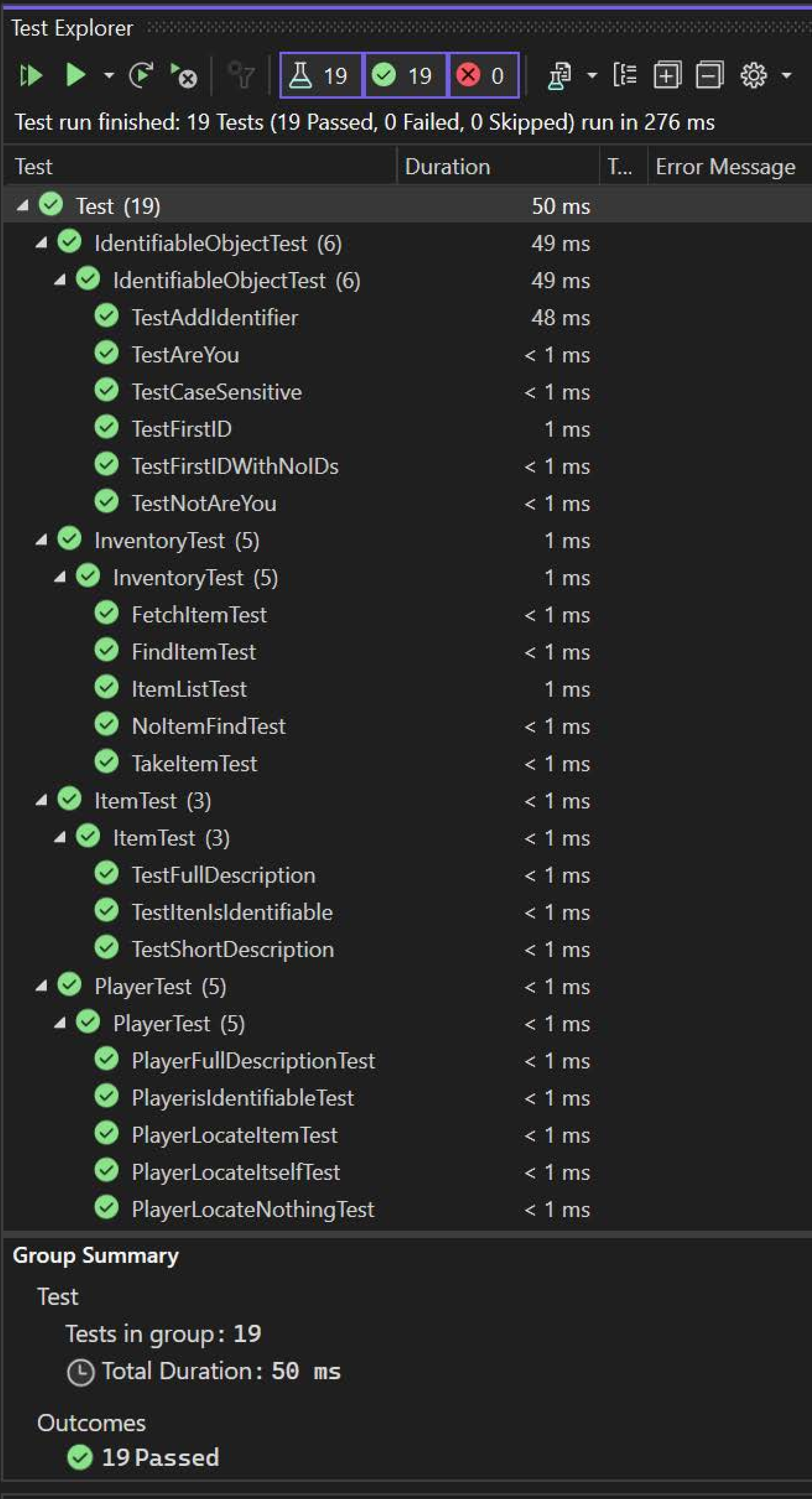
```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SwinAdventure
{
    public class Player : GameObject
    {
        private Inventory _inventory;
        string _name;
        string _desc;

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

        public GameObject? Locate(string id)
        {
            if (AreYou(id))
            {
                return this;
            }
            return _inventory.Fetch(id);
        }

        public override string FullDescription
        {
            get
            {
                return "You are " + _name + ", " + _desc + ". You are carrying: " + _inventory.ItemList;
            }
        }

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

```
using SwinAdventure;

namespace IdentifiableObjectTest
{
    public class IdentifiableObjectTest
    {
        [SetUp]
        public void Setup()
        {

        }

        [Test]
        public void TestAreYou()
        {
            IdentifiableObject obj = new IdentifiableObject(new string[]
              {"fred", "bob"});

            bool result1 = obj.AreYou("fred");
            bool result2 = obj.AreYou("bob");

            Assert.IsTrue(result1);
            Assert.IsTrue(result2);
        }

        [Test]
        public void TestNotAreYou()
        {
            IdentifiableObject obj = new IdentifiableObject(new string[]
              { "fred", "bob" });

            bool result1 = obj.AreYou("wilma");
            bool result2 = obj.AreYou("boby");

            Assert.IsFalse(result1);
            Assert.IsFalse(result2);
        }

        [Test]
        public void TestCaseSensitive()
        {
            IdentifiableObject obj = new IdentifiableObject(new string[]
              { "fred", "bob" });

            bool result1 = obj.AreYou("FRED");
            bool result2 = obj.AreYou("bOB");

            Assert.IsTrue(result1);
            Assert.IsTrue(result2);
        }

        [Test]
        public void TestFirstID()
        {
            IdentifiableObject obj = new IdentifiableObject(new string[] { "fred", "bob" });

            string firstID = obj.FirstId;

            Assert.AreEqual("fred", firstID);
        }

        [Test]
        public void TestFirstIDWithNoIDs()
        {
            IdentifiableObject obj = new IdentifiableObject(new string[] { });

            string firstID = obj.FirstId;

            Assert.AreEqual("", firstID);
        }

        [Test]
        public void TestAddIdentifier()
        {
            IdentifiableObject obj = new IdentifiableObject(new string[] { "fred", "bob" });

            obj.AddIdentifier("wilma");

            Assert.IsTrue(obj.AreYou("fred"));
            Assert.IsTrue(obj.AreYou("bob"));
            Assert.IsTrue(obj.AreYou("wilma"));
        }
    }
}
```

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

namespace InventoryTest
{
    internal class InventoryTest
    {
        Item BronzeSword = new Item(new string[] { "sword", "weapon" }, "Bronze Sword", "A simple bronze sword.");
        Item BronzeAxe = new Item(new string[] { "axe", "weapon" }, "Bronze Axe", "A simple bronze axe.");
        [SetUp]
        public void Setup()
        {

        }

        [Test]
        public void FindItemTest()
        {
            Inventory inv = new Inventory();

            inv.Put(BronzeAxe);
            inv.Put(BronzeSword);

            bool sword = inv.HasItem("sword");
            bool axe = inv.HasItem("axe");

            Assert.IsTrue(sword);
            Assert.IsTrue(axe);
        }

        [Test]
        public void NoItemFindTest()
        {
            Inventory inv = new Inventory();

            Assert.IsFalse(inv.HasItem("sword"));
            Assert.IsFalse(inv.HasItem("axe"));
        }

        [Test]
        public void FetchItemTest()
        {
            Inventory inv = new Inventory();
            inv.Put(BronzeAxe);
            inv.Put(BronzeSword);

            Item sword = inv.Fetch("sword");
            Item axe = inv.Fetch("axe");

            Assert.AreEqual(BronzeSword, sword);
            Assert.AreEqual(BronzeAxe, axe);
        }

        [Test]
        public void TakeItemTest()
        {
            Inventory inv = new Inventory();

            inv.Put(BronzeAxe);
            inv.Put(BronzeSword);

            Item sword = inv.Take("sword");
            Item axe = inv.Take("axe");

            Item SwordRemain = inv.Fetch("sword");
            Item AxeRemain = inv.Fetch("axe");

            Assert.AreEqual(BronzeSword, sword);
            Assert.AreEqual(BronzeAxe, axe);
            Assert.IsTrue(SwordRemain == null);
            Assert.IsTrue(AxeRemain == null);
        }

        [Test]
        public void ItemListTest()
        {
            Inventory inv = new Inventory();

            inv.Put(BronzeAxe);
            inv.Put(BronzeSword);

            string list = inv.ItemList;

            Assert.AreEqual("\tBronze Axe (axe)\n\tBronze Sword (sword)\n", list);
        }
    }
}
```

```
using SwinAdventure;

namespace ItemTest
{
    public class ItemTest
    {
        Item BronzeSword = new Item(new string[] { "sword", "weapon" }, "Bronze Sword", "A simple bronze sword.");
        Item BronzeAxe = new Item(new string[] { "axe", "weapon" }, "Bronze Axe", "A simple bronze axe.");
        [SetUp]
        public void Setup()
        {

        }

        [Test]
        public void TestItenIsIdentifiable()
        {
            bool SwordisIdentifiable = BronzeSword.AreYou("sword");
            bool AxeisIdentifiable = BronzeAxe.AreYou("sword");

            Assert.IsTrue(SwordisIdentifiable);
            Assert.IsFalse(AxeisIdentifiable);
        }

        [Test]
        public void TestShortDescription()
        {
            string SwordShortDesc = BronzeSword.ShortDescription;
            string AxeShortDesc = BronzeAxe.ShortDescription;

            Assert.AreEqual("Bronze Sword (sword)", SwordShortDesc);
            Assert.AreEqual("Bronze Axe (axe)", AxeShortDesc);
        }

        [Test]
        public void TestFullDescription()
        {
            string SwordFullDesc = BronzeSword.FullDescription;
            string AxeFullDesc = BronzeAxe.FullDescription;

            Assert.AreEqual("A simple bronze sword.", SwordFullDesc);
            Assert.AreEqual("A simple bronze axe.", AxeFullDesc);
        }
    }
}

```

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

namespace PlayerTest
{
    internal class PlayerTest
    {
        Player _player = new Player("Sen", "luv ur mom");
        Item BronzeSword = new Item(new string[] { "sword", "weapon" }, "Bronze Sword", "A simple bronze sword.");
        [SetUp]
        public void Setup()
        {

        }

        [Test]
        public void PlayerisIdentifiableTest()
        {
            Assert.IsTrue(_player.AreYou("me"));
            Assert.IsTrue(_player.AreYou("inventory"));
        }

        [Test]
        public void PlayerLocateItemTest()
        {
            _player.Inventory.Put(BronzeSword);

            Assert.AreEqual(BronzeSword, _player.Locate("sword"));
        }

        [Test]
        public void PlayerLocateItselfTest()
        {
            Assert.AreEqual(_player, _player.Locate("me"));
            Assert.AreEqual(_player, _player.Locate("inventory"));
        }

        [Test]
        public void PlayerLocateNothingTest()
        {
            Assert.IsNull(_player.Locate("sadaw"));
        }

        [Test]
        public void PlayerFullDescriptionTest()
        {
            _player.Inventory.Put(BronzeSword);
            string desc = _player.FullDescription;

            Assert.AreEqual("You are Sen, luv ur mom. You are carrying:\tBronze Sword (sword)\n", desc);
        }
    }
}
```



Test run finished: 19 Tests (19 Passed, 0 Failed, 0 Skipped) run in 276 ms

Test	Duration	T...	Error Message
Test (19)	50 ms		
IdentifiableObjectTest (6)	49 ms		
IdentifiableObjectTest (6)	49 ms		
TestAddIdentifier	48 ms		
TestAreYou	< 1 ms		
TestCaseSensitive	< 1 ms		
TestFirstID	1 ms		
TestFirstIDWithNoIDs	< 1 ms		
TestNotAreYou	< 1 ms		
InventoryTest (5)	1 ms		
InventoryTest (5)	1 ms		
FetchItemTest	< 1 ms		
FindItemTest	< 1 ms		
ItemListTest	1 ms		
NoItemFindTest	< 1 ms		
TakeItemTest	< 1 ms		
ItemTest (3)	< 1 ms		
ItemTest (3)	< 1 ms		
TestFullDescription	< 1 ms		
TestItenIsIdentifiable	< 1 ms		
TestShortDescription	< 1 ms		
PlayerTest (5)	< 1 ms		
PlayerTest (5)	< 1 ms		
PlayerFullDescriptionTest	< 1 ms		
PlayerisIdentifiableTest	< 1 ms		
PlayerLocateItemTest	< 1 ms		
PlayerLocateItselfTest	< 1 ms		
PlayerLocateNothingTest	< 1 ms		

Group Summary

Test

Tests in group: 19

Total Duration: 50 ms

Outcomes

19 Passed