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

namespace Iteration2
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello, World!");
        }

    }
}

```

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

namespace Iteration2
{
    public  class GameObject : IdentifiableObject
    {
        private string _description;
        private string _name;
        public GameObject(string[] ids, string name, string desc) : base(ids)
        {
            _description = desc;
            _name = name;
        }
        public string name
        {
            get { return _name; }
        }
        public string ShortDescription
        {
            get { return $"{_name} ({FirstID})"; }
        }
        public virtual string FullDescription
        {
            get { return _description; }
        }
    }

}
```

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

namespace Iteration2
{
    public class IdentifiableObject
    {
        private List<string> _identifiers;

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


        }

        public bool AreYou(string name)
        {
            foreach (string idents in _identifiers)
            {
                if (idents.ToLower() == name.ToLower())
                {
                    return true;
                }
            }

            return false;
        }

        public string FirstID
        {
            get
            {
                if (_identifiers.Count == 0)
                {
                    return "";
                }
                else
                {
                    return _identifiers.First();

                }
            }
        }

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

```

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

namespace Iteration2
{
    public class Inventory
    {
        private List<Item> _items;
        public Inventory()
        {
            _items = new List<Item>();
        }
        public bool HasItem(string id)
        {
            foreach (var item in _items)
            {
                if (item.AreYou(id))
                {
                    return true;
                }
            }
            return false;
        }
        public void Put(Item i)
        {
            _items.Add(i);
        }
        public Item Take(string id)
        {
            foreach (var item in _items)
            {
                if (item.AreYou(id))
                {
                    _items.Remove(item);
                    return item;
                }

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

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

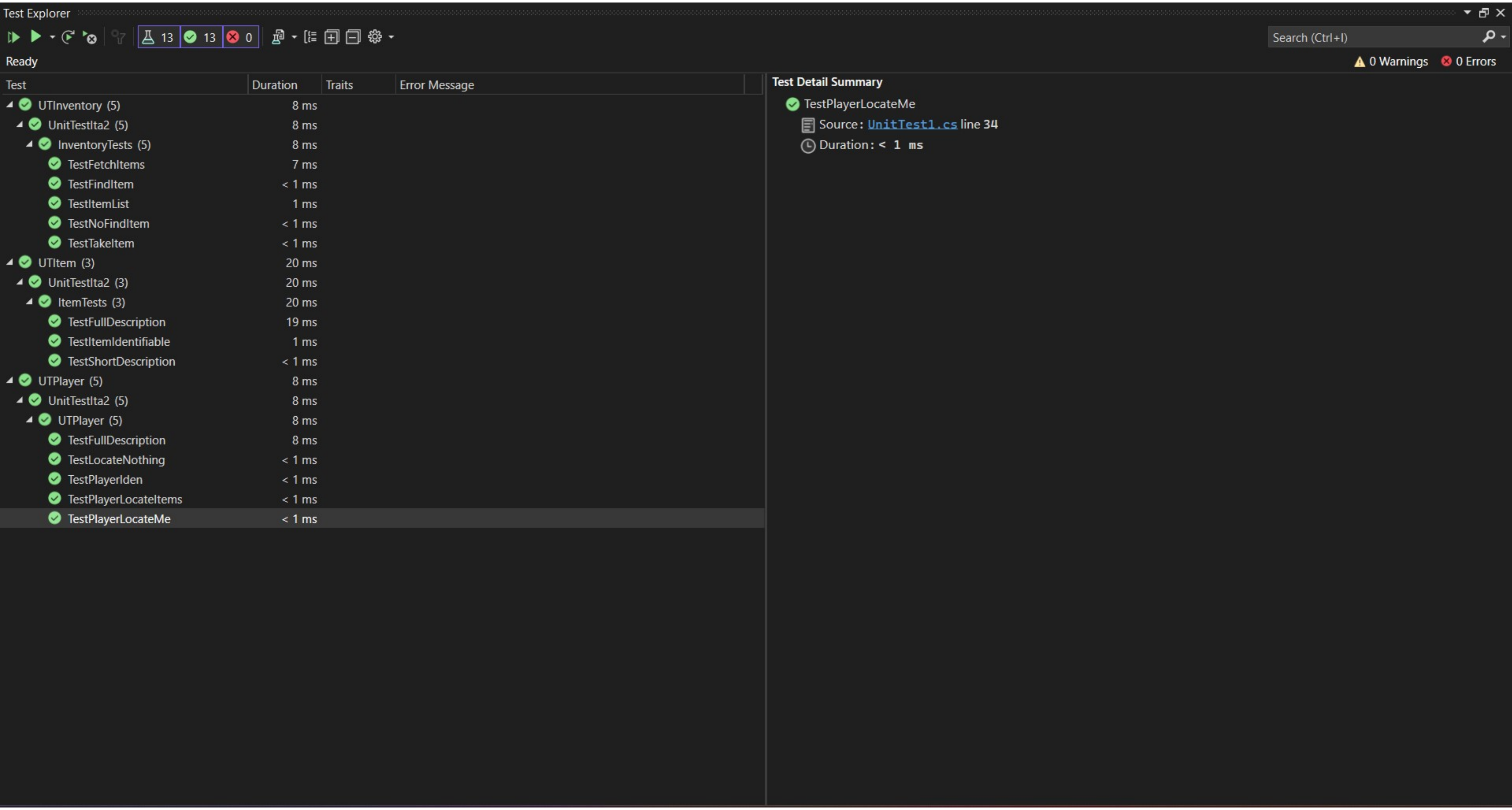
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Linq;

namespace Iteration2
{
    public class Item : GameObject
    {
        public Item(string[] idents, string name, string desc) : base(idents, name, desc)
        {

        }
    }
}

```

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

namespace Iteration2
{
    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
            {
                return _inventory.Fetch(id);
            }
        }
        public override string FullDescription
        {
            get
            {
                return $"You are {name} {base.FullDescription}. You are carrying \n" + _inventory.ItemList;
            }
        }
        public Inventory Inventory
        {
            get => _inventory;
        }

    }
}

```

```
using NUnit.Framework;
using Iteration2;

namespace UnitTestIta2
{
    public class InventoryTests
    {
        private Inventory inventory;
        private Item shovel;
        private Item pc;
        private Item sword;

        [SetUp]
        public void Setup()
        {
            Item shovel = new Item(new string[] { "shovel"," spade"}, "a shovel", "This is a shovel");
            Item sword = new Item(new string[] { "sword", " blade" }, "a sword", "This is a sword");
            Item pc = new Item(new string[] { "pc", " computer" }, "a pc", "This is a computer");
            inventory = new Inventory();
            inventory.Put(shovel);
            inventory.Put(sword);
            inventory.Put(pc);
        }

        [Test]
        public void TestFindItem()
        {
            Assert.IsTrue(inventory.HasItem("shovel"));
            Assert.IsTrue(inventory.HasItem("sword"));
            Assert.IsTrue(inventory.HasItem("pc"));
        }

        [Test]
        public void TestNoFindItem()
        {
            Assert.IsFalse(inventory.HasItem("stick"));
        }

        [Test]
        public void TestFetchItems()
        {
            Assert.NotNull(inventory.Fetch("shovel"));
            Assert.IsTrue(inventory.HasItem("shovel"));
        }

        [Test]
        public void TestTakeItem()
        {
            Assert.NotNull(inventory.Take("sword"));
            Assert.IsFalse(inventory.HasItem("sword"));
        }

        [Test]
        public void TestItemList()
        {
            string ItemList = "a shovel (shovel)\n a sword (sword)\na small computer(pc)";
            Assert.That(ItemList, Is.EqualTo(ItemList));
        }

    }
}

```

```
using NUnit.Framework;
using Iteration2;

namespace UnitTestIta2
{
    public class ItemTests
    {
        Item shovel = new Item(new string[] { "shovel" }, "a shovel", "This is a shovel");
        Item sword = new Item(new string[] { "sword" }, "a sword", "This is a sword");


        [SetUp]
        public void Setup()
        {

        }

        [Test]
        public void TestItemIdentifiable()
        {
            var result = shovel.AreYou("sword");
            Assert.IsFalse(result); //Item cannot be defined

            var result2 = shovel.AreYou("shovel");
            Assert.IsTrue(result2);

        }

        [Test]
        public void TestShortDescription()
        {
            Assert.AreEqual(shovel.ShortDescription, "a shovel (shovel)"); //Description Correct!
            Assert.AreNotEqual(shovel.ShortDescription, "This is a shovel"); //Testing short with long Description showing they are not Correct!
        }

        [Test]
        public void TestFullDescription()
        {
            Assert.AreEqual(shovel.FullDescription, "This is a shovel"); // Full Description is Correct!
            Assert.AreNotEqual(shovel.FullDescription, "a shovel (shovel)"); //Full Description is not Correct!

        }
    }
}
```

```
using NUnit.Framework;
using Iteration2;
using NuGet.Frameworks;

namespace UnitTestIta2
{
    public class UTPlayer
    {
        private Player player;
        [SetUp]
        public void Setup()
        {
            player = new Player("fred", "the mighty programmer");
            player.Inventory.Put(new Item(new string[] { "shovel", "spade" }, " a shovel", "This is a fine shovel"));
            player.Inventory.Put(new Item(new string[] { "sword", "blade" }, " a sword", "This is a fine sword"));
            player.Inventory.Put(new Item(new string[] { "pc", "computer" }, " a computer", "This is a fine computer"));
        }

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

        [Test]
        public void TestPlayerLocateItems()
        {
            Assert.IsTrue(player.Locate("shovel").AreYou("shovel"));
            Assert.IsTrue(player.Locate("sword").AreYou("sword"));
            Assert.IsTrue(player.Locate("pc").AreYou("pc"));
        }
        [Test]
        public void TestPlayerLocateMe()
        {
            Assert.IsTrue(player.Locate("me").AreYou("me"));
            Assert.IsTrue(player.Locate("inventory").AreYou("inventory"));
        }
        [Test]
        public void TestLocateNothing()
        {
            Assert.IsNull(player.Locate("stick"));
        }
        [Test]
        public void TestFullDescription()
        {
            string pFullDescription = "You are fred the mighty programmer. You are carrying \n a shovel (shovel)\n a sword (sword)\n a computer (pc)\n";
            Assert.That(player.FullDescription,Is.EqualTo(pFullDescription));
        }
    }
}
```



Ready ⚠ 0 Warnings ⊗ 0 Errors

```
Test                                  Duration  Traits  Error Message
◢ ✓ UTInventory (5)                       8 ms
  ◢ ✓ UnitTestIta2 (5)                    8 ms
    ◢ ✓ InventoryTests (5)                8 ms
        ✓ TestFetchItems                  7 ms
        ✓ TestFindItem                  < 1 ms
        ✓ TestItemList                    1 ms
        ✓ TestNoFindItem                < 1 ms
        ✓ TestTakeItem                  < 1 ms
◢ ✓ UTItem (3)                           20 ms
  ◢ ✓ UnitTestIta2 (3)                   20 ms
    ◢ ✓ ItemTests (3)                    20 ms
        ✓ TestFullDescription            19 ms
        ✓ TestItemIdentifiable            1 ms
        ✓ TestShortDescription          < 1 ms
◢ ✓ UTPlayer (5)                          8 ms
  ◢ ✓ UnitTestIta2 (5)                    8 ms
    ◢ ✓ UTPlayer (5)                      8 ms
        ✓ TestFullDescription             8 ms
        ✓ TestLocateNothing             < 1 ms
        ✓ TestPlayerIden                < 1 ms
        ✓ TestPlayerLocateItems         < 1 ms
        ✓ TestPlayerLocateMe            < 1 ms
```

Test Detail Summary

✓ TestPlayerLocateMe

Source: UnitTest1.cs line 34

Duration: < 1 ms